THE ROLE OF HETEROGENEITY IN RHYTHMIC NETWORKS OF NEURONS

A Thesis
Presented to
The Academic Faculty

by

Michael S. Reid

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in Electrical and Computer Engineering in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
May 2007

THE ROLE OF HETEROGENEITY IN RHYTHMIC NETWORKS OF NEURONS

Approved by:

Dr. Stephen P. DeWeerth, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Robert J. Butera, Jr.
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Robert H. Lee
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Paul E. Hasler
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Astrid Prinz
Biology Department
Emory University

Date Approved: December 18, 2006

For my family—

Lucy, Michael, and Steven.

ACKNOWLEDGEMENTS

Suggesting that my time at the Georgia Institute of Technology was challenging would be an understatement—obtaining a Ph.D. from Georgia Tech's elite School of Electrical and Computer Engineering was a formidable task, to say the least. My lifestyle also changed dramatically during these years. In 1999, I vacated a 500-square-foot apartment, in which pots hung from the ceiling, on Manhattan's Upper West Side after working on Wall Street for five years. I traded subway rides for interstate drives, corporate life for academic life, suits for shorts, and snow for pollen. This is a time of reflection for me as this chapter of my life ends. In doing so, I would like to recognize the following people:

I would like to thank Dr. Fallaw Sowell from Carnegie Mellon University for all of his guidance—I always know where to go when I have a tough question to ask.

I would like to thank my childhood friend Rob Codell, who always gave me a unique perspective on a wide variety of topics—his thoughts still influence me considerably.

I would like to thank my sister, Mari Marlow, and my close friends, Ken Mazurek, Tim Skaryd, and Mike Weeks—these are the people that I can always count on and that I trust for support and guidance. I also want to thank Ted and Jennifer Hansen for their support and friendship from the time we moved to Atlanta—we went from late-night card games to late-night feedings but we wouldn't have changed a thing.

I would like to thank the following past and present members of the Laboratory for Neuroengineering: Chuck Wilson, Tina Hudson, Alex Bragg, Mario Simoni, Edgar Brown, David Lin, Jason Meeks, Mara Carey, Richard Blum, Michael Sorensen, Chris Duffy, Abhishek Bandyopadhyay, Paul Smith, Paul Garcia, Luke Purvis, Guillermo Serrano, Farhan Adil, Shane Migliore, Jevin Scrivens, Kyla Ross, Jim Ross, Bobby Brooke, Shawn O'Connor, Ivan Raikov, William Gerken, Murat Sekerli, Amanda Preyer, Simeon Sessley, Carrie Williams, Kate Williams, Kartik Sundar, Brian Paul Degnan, Scott Buscemi, Randy Weinstein, Chris Church, Stefan Clemens, Bryan Williams, Amber Burris, and Jon Hall. These

are incredibly intelligent and talented people, and without a doubt, I would not have succeeded without the help and support of all of them. Aside from work, I have enjoyed the lunches, the racquetball and tennis matches, the baseball games and catches, and the countless debates and bets (all of which I won ...).

I would like to the thank those affiliated with the Emory University Neuroscience program, including Dr. Gennady Cymbalyuk and Dr. Astrid Prinz.

I would like to thank the following faculty in ECE: Dr. John Peatman, Dr. J. Alvin Connelly, Dr. Phillip Allen, Dr. Paul Hasler, and the late Dr. John Uyemura. These professors helped to establish my early impressions of graduate school at Georgia Tech, which will last my lifetime.

I would like to thank Dr. William Sayle, who gave me the opportunity to teach courses in ECE, including those usually reserved for full-time faculty. His trust in me was very much appreciated and will not be forgotten.

I would like to thank the faculty of the Laboratory for Neuroengineering, especially Dr. Robert Butera and Dr. Robert Lee, whose guidance was indispensable.

I would like to thank my advisor, Dr. Stephen DeWeerth, who took a chance on this older married guy, who was going back to school after working in industry for many years. I appreciate the opportunities that he gave to me, in both research and teaching, and the options that will now be available to me as a result. I clearly see the world differently after my years under his guidance. Under his instruction, he shaped and greatly improved my ability to communicate ideas, to think critically, and to analyze results—skills that characterize me not only as a researcher, but also as a person.

I will not be defined, however, by my years at Georgia Tech; instead, success for me will be determined by my ability to parent my two boys, Michael and Steven. I would like to thank my wife, Lucy, not only for her unwavering support of my pursuit of this doctoral degree, without which I would certainly not have finished, but also for giving me these two wonderful gifts. I would also like to thank her for being an incredible mother to these two very lucky boys—lucky because they get to say that they are her sons. Lucy—I could not have done this without you. Saying thanks would be another understatement.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Engineers often view variability as undesirable and seek to minimize it, such as when they employ transistor-matching techniques to improve circuit and system performance. Biology, however, makes no discernible attempt to avoid this variability, which is particularly evident in biological nervous systems whose neurons exhibit marked variability in their cellular properties. In previous studies, this heterogeneity has been shown to have mixed consequences on network rhythmicity, which is essential to locomotion and other oscillatory neural behaviors. The systems that produce and control these stereotyped movements have been optimized to be energy efficient and dependable, and one particularly well-studied rhythmic network is the central pattern generator (CPG), which is capable of generating a coordinated, rhythmic pattern of motor activity in the absence of phasic sensory input. Because they are ubiquitous in biological preparations and reveal a variety of physiological behaviors, these networks provide a platform for studying a critical set of biological control paradigms and inspire research into engineered systems that exploit these underlying principles. We are directing our efforts toward the implementation of applicable technologies and modeling to better understand the combination of these two concepts—the role of heterogeneity in rhythmic networks of neurons. The central engineering theme of our work is to use digital and analog platforms to design and build Hodgkin–Huxley conductance-based neuron models that will be used to implement a half-center oscillator (HCO) model of a CPG. The primary scientific question that we will address is to what extent this heterogeneity affects the rhythmicity of a network of neurons. To do so, we will first analyze the locations, continuities, and sizes of bursting regions using single-neuron models and will then use an FPGA model neuron to study parametric and topological heterogeneity in a fully-connected 36-neuron HCO. We found that heterogeneity *can* lead to more robust rhythmic networks of neurons, but the type and quantity of heterogeneity and the population-level metric that is used to analyze bursting are critical in determining *when* this occurs.

CHAPTER I

INTRODUCTION

The neuroscience community is still confounded by neuromuscular disorders, including spinal-cord injuries, while those that suffer from these afflictions continue to wait for cures. Neuroscientists, however, have been aided by progress in two complementary goals—the advancement of scientific understanding and the improvement of engineering technology. These achievements have resulted in remarkable contributions in the study of cellular behavior, including the formulation over 50 years ago of the Hodgkin–Huxley (HH) model, which ranks as one of the most significant conceptual breakthroughs in this field. Notwithstanding this progress and motivated by elevated awareness and increased spending that has been driven by people such as the late Christopher Reeve, scientists continue to search for cures while they lack understanding of not only the pathology but also the healthy physiology that underlie movement. To better understand and address the needs of motor-control diseases, we are directing our efforts toward the implementation of tools—applicable technologies and modeling—that will allow us to study the critical pattern-generating networks of neurons that produce and control rhythmic movements.

We want to make predictions about neurobiological systems in which the answers to our scientific questions would be difficult, if not exceedingly difficult, to answer through experimentation with the actual system because invasive human experiments are not possible and animal experiments are controversial, difficult, and expensive (in terms of time and cost). As a result, theoretical mathematical models offer an effective platform to examine our questions of interest, and we will develop artificial physical systems with these models to undertake our complementary scientific and engineering goals. In our work, our real systems of interest involve stereotypical, open-loop, neural oscillators. In particular, a central pattern generator (CPG) is a network of neurons that is capable of generating a complex, coordinated, rhythmic pattern of motor activity in the absence of phasic sensory

input [62] and reveals a variety of physiological behaviors [79]. To model the CPG, we will use a half-center oscillator (HCO), which is typically represented as two systems of neurons that reciprocally inhibit each other; these two systems were originally termed *half-centers* [7]. We will introduce a variety of forms of heterogeneity (intrinsic, parametric, and topological) in a variety of neural implementations (digital and analog) and study the responses to determine the extent that heterogeneity is functionally advantageous.

The central engineering theme of our work is to design and build a rhythmic, pattern-generating system to study the role of heterogeneity in biological networks, and the primary scientific question that we will address is to what extent this heterogeneity increases measures of output robustness in the network. To accomplish this goal, we will model a CPG using a HCO formed from a population of heterogeneous neurons, which are pervasive in biological systems [101] [39]. We will use a HCO because it is ubiquitous in biological preparations, including the leech heartbeat generator [86] [87] and a subnetwork of the stomatogastric pyloric rhythm in the lobster [83]. CPGs have been well examined in invertebrates, and one of the best-studied CPGs is involved in lamprey swimming [62]. We realize, however, that we must be careful with extending our findings to the human case because an open question still remains as to whether CPGs exist in human spinal cords [75]. We will model the neurons using the HH formalism [55], which represents a good tradeoff between complexity and dynamical output. This conductance-based neuron model ensures biologically realistic dynamics, an important characteristic because a consideration of the physiological details of a real neural system is likely to lead to important clues to its function [4]. To answer our primary scientific question, we will study small networks of neurons, rather than more-controllable individual neurons, because the redundancy of the population potentially leads to greater robustness across the network.

1.1 Background and Significance

To understand the role of heterogeneity in rhythmic networks of neurons, we must identify the modeling, technological, design, and scientific topics that are relevant to our research and how our research will complement the previous work in those areas. For example, with

our engineering work, we must show the progress that we made with our neuromorphic and field-programmable gate array (FPGA) designs compared to previous implementations. In addition, we must specifically address how the science of our work (the study of heterogeneity) will help to address open questions in the literature.

1.1.1 Biological Control of Rhythmic Movements

CPGs, neural circuits that can generate periodic rhythmic bursting, are involved in sensory, motor, and cognitive tasks in a wide variety of animals [79]. In 1911, Brown conceived of the mechanism of the simplest emergent rhythm, an HCO that generated bursting behavior in the mammalian spinal cord [7]. The HCO is a useful model to represent anti-phasic bursting in neural systems because of its simplicity and symmetry. The output of a CPG drives muscle movements, and relationships between this neural control and neuromuscular behavior have been studied in simple model systems. For example, simple first-order models have been useful for early studies of complex movements [49] [48] [34] [29] [11] [12].

Recent experimental and theoretical work has shown that oscillatory processes are important in memory formation [72], olfaction [68], and visual processing [42]. CPG circuits are also involved in the generation of various rhythmic and stereotyped movements such as walking, breathing, swimming, and chewing [79]. Behaviors that are controlled by CPG circuits can also be characterized by their timing—some of these motor-control movements are ongoing, such as the leech heartbeat generator [86] [87], and some can occur for short or long periods of time, such as swimming in lampreys and fish [119] [131].

In more complex CPGs, the combined activity and interaction of neural populations generate these rhythms. In addition, significant variability in the cellular properties of the neurons that form a CPG may exist [101] [39], and recent studies have examined potential functional roles for such neural heterogeneity. Using a model of the pre-Bötzinger complex (PBC), the authors showed that the range of input parameters in which a heterogeneous network produced synchronous rhythms was much larger for the network than for a single neuron whose parameters were set to their mean values [9]. An open and interesting question that we will explore is whether heterogeneity in a biological network increases system

stability and controllability. To do so, we must first examine forms and consequences of heterogeneity.

We must be able to define and quantify *heterogeneity* before we can study its influence on a system. After an exhaustive literature search, we have concluded that its definition is open to a wide interpretation. Table 1 summarizes some of the general quantitative uses of the term and some of the uses specific to synchrony. Note that in the second section of the table, only the last three references conclude that heterogeneity has a negative impact on synchrony, and the first and last references appear to be in direct contradiction, while using many of the same references (*c.f.*, [67]).

Examples of the qualitative nature of heterogeneity are also apparent in the following references with no quantitative justification: width and degree of heterogeneity [129]; mild, weak, small, large, strong, and broad heterogeneity [13]; weak heterogeneity [63]; moderate and small heterogeneity [120]. Some authors have used the term *heterogeneity* as the first word of their article titles, but used the term sparingly in their text [133] [65] [117]. Other authors used *heterogeneity* in their article title [58] or as an article keyword [98], but like the other authors cited, they seemed to be using the term simply as a synonym for "different," "dissimilar," "diverse," "mixed," or "varied" (and their variants). Authors have also used such terms as "inhomogenous" (and its variants), "nonidentical," "intrinsic disorder," and "parameter variation" to refer to heterogeneity. To avoid ambiguous descriptions, we always quantified both the heterogeneity that we imposed on our system and the output changes that occurred as a result of the applied heterogeneity.

1.1.2 Modeling Neural Systems

The reduced model neuron used in our work can yield new functional neurobiological relationships because of its classification as a *realistic model* [2], one that is based on the actual anatomy and physiology of the nervous system and can thus be used as a tool to discover unknown relationships between the structure and function of the nervous system [1] [3]. Alternatively, a *demonstration model* provides support for a particular preexisting functional point of view [1]. In fact, a well-conceived program of modeling can result in experimental

Table 1: Sampling of heterogeneity references separated by type—general and synchronous.

First Author	Ref	Comment
General Heterogeneity Usages		
Butera	[9]	refers to heterogeneity in intrinsic properties, synaptic conductances, and synaptic connectivity
Karbowski	[63]	defines heterogeneity as the differences in the intrinsic oscillator frequencies
Chow	[13]	states that heterogeneity is in the intrinsic firing frequency of the individual neurons
Tiesinga	[112]	includes heterogeneity in the applied current and claims that this current heterogeneity represents a variation in intrinsic neuronal properties
Miller	[84]	determines synaptic connection strengths from Gaussian distributions while referring to the resulting weights as *heterogeneous tuning curves of memory activity* and then studies the stability to this heterogeneity within a population of neurons
Laurent	[69]	describes neurons that showed heterogeneity in their outward currents; also describes heterogeneity in other contexts such as gender-specific functional heterogeneity and heterogeneity in the action potential firing properties
Valdés	[116]	differentiates between classical, heterogeneous, and fuzzy heterogeneous neurons (neural networks) in which the last two types of neurons may accept a mixture of real, qualitative, and fuzzy quantities, possibly with missing information, as opposed to classical neurons where inputs are continuous real-valued quantities
Heterogeneity Usages Specific to Synchrony		
Chow	[13]	claims that heterogeneity can greatly reduce synchrony in networks of phase-coupled oscillators and that it's well known
Tiesinga	[112]	claims that heterogeneity (and noise) lead to a reduction in synchronization
Wang	[120]	expects intuitively that network synchrony cannot be globally maintained if individual neurons display very different intrinsic oscillation frequencies
White	[129]	believes that increasing the heterogeneity reduces the size of the synchronous region
Tsodyks	[114]	shows that for globally coupled oscillators with neuron-like pulse interactions, inhomogeneity in the local frequency creates instability in the phase-locked state
Butera	[9]	illustrates that intrinsic parameter heterogeneity makes rhythmic bursting more robust
Grimm	[44]	indicates in the article title that a population of neurons contributes to the robustness of a CPG against parameter variation
Wiesenfeld	[130]	states that populations of coupled nonlinear oscillators can spontaneously synchronize to a common frequency, despite differences in their natural frequencies, and that it's well known

tests that identify basic cellular mechanisms of neuronal network function [10]. Consequently, we expected the results from testing with the reduced model neuron to replicate physiological data, direct new experimental efforts, and provide predictive capability.

The reduced model neuron used in our work was a single-compartment HH model. We use the term *reduced* because it requires only a limited number of ionic channels and parameters. This reduction provides a good balance between complexity and dynamical output—that is, the membrane dynamics are represented faithfully with biologically relevant dynamics. This nonlinear, multi-input model is minimized to that necessary to produce bursting behaviors (*i.e.*, has both slow and fast time constants to produce the underlying oscillations and spiking, respectively) and has been extensively verified through experimental studies [8] [9] [18] [17]. This endogenous burster model also exhibits robust bursting behaviors such that single-neuron data can be gathered easily.

Although the study of the elements of a computational system is an essential step in understanding and controlling the system organization, the system complexity derives from the multitude of ways in which a large collection of these components can interact [81]. As a result, we maintained a proper balance between the work with a single neuron (the component) and the work with a network of neurons (the system). Our work with circuit- and system-level neural modeling was inspired in part by that of the neuromorphic community.

1.1.3 Neuromorphic Architectures

The field of neuromorphic engineering was pioneered by Carver Mead in the 1980's. It involves the design and fabrication of hardware-based artificial neural systems, predominantly using custom-designed analog circuits, that are inspired by circuit architectures found in biological nervous systems and that impose physical constraints, like those in their biological counterparts. Neuromorphic analog very large-scale integrated (aVLSI) circuits [81] [73] are typically used as the building blocks for modeling these neurophysiological systems, which provide a platform for developing real-world, real-time, large-scale, hierarchical, parallel, robust, and complex engineered systems. These systems are also used as neural modeling

tools that give immediate feedback and intuition to the experimenter, thus allowing the discovery of fundamental principles through the immediate correlation of parameter variation to observation of behavioral changes. The parallel and hierarchical design also allows aVLSI chips to be scaled easily into large-scale systems.

A misconception in the field of electronics is that digital circuits are replacing analog ones, but analog circuits have a variety of current applications, including cellular telephones, fiberoptics, and video systems. Note that because rise times, delays, and settling times are analog properties of nominally digital circuits, *all* circuits are analog, even if they are used as digital building blocks. Because real systems are asynchronous and analog in nature, asynchronous analog circuits (as opposed to digital circuits) are well suited for interfacing to real systems. In addition, analog circuits use less power [100] and are also much smaller. For example, consider the silicon area that is required for a digital multiplier as opposed to an analog one or consider that an n-bit digital output requires n wires. Exact, digital results are not necessarily required in the study of rhythmic pattern-generating systems; instead, the focus can be on the qualitative aspects of behavior. These lightweight, low-power, portable aVLSI chips also provide engineers and scientists with the ability to power an autonomous robotic application because a tethered cable is not necessary, and an implantable, neural-prosthetic design is possible because of the advantageous size and power-consumption requirements.

Although the special-purpose neuromorphic aVLSI architectures have been applied primarily to sensory-processing tasks such as those related to the retina [77] and cochlea [74], this technology is equally well suited to the modeling of biological motor systems [91]. These circuits have also been used for creating neurally inspired motor control systems including rhythmic motor-pattern generation [92] [91] [108], oculomotor feedback [56] [20], higher-level (cortical) processing [45], and muscular control [57]. aVLSI chips can also easily be interfaced to mechanical actuators to create a system that is able to control external systems or one that interacts with the environment such as a mobile robot [21].

To study neural dynamics, neuromorphic engineers have also designed silicon neurons using aVLSI circuits. These aVLSI circuits have been used to implement neurons based on

HH and other formalisms. These silicon model neurons have been inspired by the classical design of Mahowald and Douglas [76] [24]. In later improvements to that design using the HH formalism [55], one model used less circuitry and fewer parameters that improved the spiking characteristics [95], and another model used a multi-chip, BiCMOS design [26]. Another model using the HH formalism was developed by Simoni [105] and is based on a mathematical model of the leech heart interneuron [86]. This design represented further improvements in the neural dynamics, circuit size, and interfacing capabilities and is the basis for the silicon model neuron that was used in our work. In addition, this silicon-neuron architecture is versatile because of its ability to implement a variety of conductance-based neuron models, and in one of the first attempts to show the ability to scale the classical silicon model neuron, we used it to implement a network of silicon model neurons. These and many other examples illustrate the progress that has been inspired by the first silicon neuron [76]. A great opportunity, however, still exists to demonstrate the applicability of these architectures to a broad set of problems.

Analog VLSI circuits have also been used to implement other neural models in silicon. An adaptive CPG in an aVLSI chip using integrate-and-fire (IF) neurons [35] [36] was used to control a running robot leg by Lewis et al. [70]; in fact, Mead first described similar neural functionality with the self-resetting neuron [81]. Selverston and Abarbanel used the Hindmarsh-Rose (HR) model as the basis for building silicon model neurons that could then be integrated into biological circuitry [103]. Patel et al. developed a silicon model neuron using the Morris-Lecar (ML) model that simplified the neural dynamics to only two state variables and was used to study slow oscillations [15] [90]. Linares et al. used the FitzHugh-Nagumo (FHN) model to emulate a small biological system in silicon [71]. Finally, Farquhar designed a circuit whose basis is not a mathematical neural model but rather uses the correlation between semiconductor and biological physics [31] [30].

Unlike a computer model, a network of one or one thousand silicon neurons operates in the same time. Also, the micron-sized transistors etched in silicon that form the neuronal circuits consume very little power and have potential usefulness in the field of autonomous robots in which size, portability, and power are important. In addition to these macro-

and micro-scalability issues, previous studies have shown that real-world silicon circuits can stably reproduce many of the dynamical regimes that exist in a theoretical model; for example, bifurcation analysis of a mathematical model led to accurate predictions of oscillatory behavior observed experimentally with an aVLSI system [15].

Because the conductances, currents, and time constants of silicon model neurons are on a biological scale (although the membrane voltage has to be scaled), hybrid neural-electronic systems are possible in which real-world, real-time interfacing (implemented with bidirectional communication) between the silicon model neurons and living neurons requires only standard neurophysiological equipment [107]. Notwithstanding the importance of a physical implementation using these neuromorphic architectures, we understand the limitations of using custom aVLSI technology for building neural systems, such as: (1) Design decisions must be well-defined prior to implementation and thus may lead to inflexible, task-specific circuits. (2) Design cycles are long. (3) Model state variables are imprecise. Recognizing these constraints, we are also pursuing other alternatives to neural modeling. For example, we have used FPGAs to outperform computers in simulating relatively simple neural models [41]. Other results suggest that FPGAs could provide even larger performance gains with more complex models and also increase the speed of automated parameter searches [123] [110]. As a result, we are not working in isolation with our silicon neurons; instead, we appreciate and utilize other neural modeling technologies, specifically FPGA neural implementations, in our study of neurobiological systems.

1.1.4 FPGA Implementations of Neural Models

The FPGA is a digital semiconductor device used with custom applications that comprises arrays of reconfigurable elements (*e.g.*, binary logic gates, lookup tables), fixed-logic resources (*e.g.*, multipliers, multiplexors, memory modules), and interconnects. FPGAs[1] are poised to make a significant impact in the neural modeling community. These devices combine the reconfigurable nature of general-purpose computers with the performance of

[1]Most of the material in this section was taken from a paper by Weinstein, Reid, and Lee that was accepted by IEEE Transactions on Neural Systems and Rehabilitation Engineering [122].

dedicated-hardware processing, allowing the realization of complex neural models with performance exceeding computers by a factor of 10 to 100 [41] [124]. Due to the difficulty in designing and implementing complex systems, FPGAs have traditionally been held within the realm of dedicated hardware engineers using specification languages such as Verilog or VHDL. A new generation of development environments based in MATLAB-SIMULINK and LabView (National Instruments, Austin, TX) have made FPGA development more accessible, but significant domain knowledge is still required.

FPGAs have been previously shown to be a high-performance platform for neural-modeling applications [41] [124] [78] [111]. Given this fact, our recent research has been focused on addressing and reducing the development time. FPGA implementation is not simply an enumeration of the model's equations in a programming language. Instead, a variety of transformations must be made to work within the confines of the FPGA architecture. These transformations include numerical-precision conversion, operation substitutions, and resource-constrained expression folding. Although these architectural confines can be overcome, they nonetheless frustrate and extend the design process and often introduce errors throughout the design flow. For our work, we used a new process (described in Section 2.3.8 and Section 2.3.9) that utilized auto-generated scripts and run-time interaction tools to further enhance the use of FPGAs as neural-modeling platforms. This process will be demonstrated via the construction of a fully interconnected population of PBC neuron models, and this FPGA model neuron will be used to answer scientific questions related to the impact of heterogeneity on conductance-based rhythmic populations of neurons.

To date, most of the published FPGA work with neurons has been in the area of neural networks, although some biologically realistic work includes an implementation of an electronic cochlear filter [60] and a comparison of biological auditory perception neurons [82]. A neuron model implemented with an FPGA is less complex, more stable, more accurate, and more flexible than one implemented in silicon. In addition, the design time required to implement a network of fully connected neurons, the central theme of the work described in this document, is reduced. For modeling purposes, the real-time nature of silicon is actually a disadvantage for parameter space mappings, whereas the FPGA can run much faster than

real time. In fact, a generalized, scalable FPGA-based architecture was shown to compute faster than a general-purpose computer [41].

The implementation of a model, using conventional software modeling tools or an FPGA, can be divided into a structural design phase and a parameter tuning phase. For mechanistic models, the structural design phase consists of building components (*e.g.*, ion channels, synapses) that are then combined into a neural-membrane model [61] [54]. This process is repeated, generating multiple neurons and synapses in a population model. Morphological models are structurally grown by linking multiple adjacent membrane models via coupling conductances [102] [118]. Once constructed, the resulting model must be tuned for the desired outputs. Parameter sets can be found by automated search tools or by hand tuning. Often, experimentation on the model requires frequent adjustment of parameter sets. Structural changes, such as insertion or deletion of ionic currents, are less frequent.

First-generation FPGA implementations of neural models [41] were "handcrafted," one-of-a-kind designs that required weeks to months for each design iteration and hours for parameter updates. This generation of models involved direct translation from a SIMULINK-based continuous-time, variable-time step, floating-point precision model into a System Generator (Xilinx, San Jose, CA) blockset-based discrete-time, fixed-time step, fixed-point precision model. This process was enhanced through pipelining, a methodology by which multiple models can be executed simultaneously utilizing the inherent delays of the system [41]. This model was groundbreaking in that it first introduced FPGAs to mechanistic neural modeling, but it required significant effort for design generation and iterations. This approach, based directly off the SIMULINK model, however, was not optimal for hardware generation; it utilized excessive resources for limited performance and offered little flexibility once implemented. Furthermore, this generation limited the complexity of models that could be implemented. Specifically, this design methodology lacked a clear paradigm for model generation and required a new development cycle as the model changed.

Based on the lessons learned from the handcrafted designs, second-generation FPGA implementations first formalized [124] then generalized [121] the design process. This "engineered" approach made a clear distinction between the computational components (the

data-path) and memory-based components (the states). In short, for each simulated time step of the model, the data-path computed the next state of each differential equation, the results of which were then stored back into memory (implemented as a shift register). This distinction between the computational and memory components eliminated the need for reworking the data-path (*e.g.*, when changing the number of neurons). Furthermore, the memory-based components were developed independently according to a set of rules and heuristics. Rules for choosing operation latency (the number of cycles required for computing a result per operation) and precision were suggested, easing the design process.

Structural modifications with the handcrafted approach required weeks to months compared to days for the engineered approach. While the second-generation approach resulted in significant time savings over the first-generation approach, further improvements were desirable. Parameter adjustment effort remained somewhat constant, often taking hours for a recompilation of the system. The third-generation approach to developing FPGA-based neural models, the "assisted" approach (described in Section 2.3.8 and Section 2.3.9), offered a number of advantages over the previous architectures, for both the construction of the model and the interaction with the implemented model. This approach consisted of design tools that offered the ability to make many structural changes within hours and parameter adjustments on-the-fly.

1.2 Engineering and Scientific Aims

In our pursuit to study the role of heterogeneity in rhythmic networks of neurons, we achieved a number of engineering and scientific goals. These included building models using a variety of implementation platforms and introducing heterogeneity using both analog circuits and FPGA programming. The science that we sought began with single-neuron studies using these platforms and culminated with a fully-connected HCO FPGA implementation in which heterogeneity was completely controlled and was introduced in parametric and topological forms.

We implemented a reduced model neuron using an analog technology and a digital technology. In this design aim, we used custom silicon circuits to build an analog model

and an FPGA, ported from MATLAB-SIMULINK (Mathworks, Cambridge, MA), to build a digital model. We used an analog VLSI platform because of its applicability in such fields as autonomous robotics and neural prosthetics that result from its lightweight, low-power, and portable design and because it is naturally suited for these types of large-scale, hierarchical, and real-world systems. A neuron model implemented with an FPGA, however, has more flexibility, allowing a modeler to study a broader set of conditions. In addition, the FPGA is less complex and more stable, and the design time required to implement a fully connected HCO, the central theme of our work, was reduced with an FPGA implementation. As a result, we believe that the FPGA model neuron allowed us to better understand how synchrony is generated, maintained, and eliminated in rhythmic networks of neurons, and because of this understanding, we were able to focus more on scientific questions and less on engineering design issues.

We used single-neuron models to analyze the locations, continuities, and sizes of bursting regions in addition to bursting characteristics such as period, duty cycle, and spike frequency. Using the MATLAB-SIMULINK model neuron and the silicon model neuron, we used two disparate methods to examine the role of parameter variability on the locations and continuities of bursting regions. We hypothesized that the bursting points found in these multi-dimensional parameter spaces lie within a single contiguous bursting region. We also analyzed the effects of heterogeneity in a network of isolated silicon model neurons by studying the mappings between the input space (model parameters) and output space (time- and voltage-based measures).

The ultimate goal of our work was to analyze the effects of heterogeneity in rhythmic networks of neurons. To do so, we first used the FPGA to configure a homogeneous, fully connected, symmetrical HCO. We extended our analysis of bursting regions by determining how the sizes of these regions were changed by two population variables (network size and coupling strength) in different regions of intrinsic parameter space. The results of these studies were used as benchmarks for the analysis of the data that was obtained with the heterogenous configurations. For these network configurations, we hypothesized that the heterogeneity imposed on a rhythmic network of neurons would result in positive

changes in measures of output robustness. Included in our hypothesis was a belief that, for a given level of heterogeneity, regions in parameter space will exist in which individual neurons will not burst endogenously, but because of population effects, will exhibit anti-phasic bursting when connected in a HCO configuration. Using the FPGA to configure a variety of these heterogeneous HCOs, such as ones with synaptic heterogeneity to study the role of coupling (weak versus strong) and connections (sparse versus dense), we performed a series of experiments to examine how network rhythmicity was influenced by heterogeneity.

CHAPTER II

IMPLEMENTATION OF NEURON MODELS

The object of our research is to study the role of heterogeneity in systems that emulate biological networks of neurons. Engineers often view variability as undesirable and seek to minimize it, such as when they employ transistor-matching techniques to improve circuit and system performance. Biology, however, makes no discernible attempt to avoid this variability, which is particularly evident in biological nervous systems whose neurons exhibit marked variability in their cellular properties [101] [39]. This heterogeneity has also been shown to have positive effects on biological circuits and systems by contributing to the robustness in the network [64]. While maintaining a proper balance between science and engineering, we developed this research through the design, experimentation, and analysis discussed in this document. In this chapter, we discuss the implementation of the conductance-based neuron model using both digital and analog platforms. In Chapter 3, we will analyze the locations, continuities, and sizes of bursting regions using single-neuron models using a computational model and a physical model. In Chapter 4, we will analyze the roles of a variety of types of heterogeneity imposed on rhythmic networks of neurons and will quantify the resulting changes to determine how heterogeneity affects the robustness of the network.

2.1 Reduced Model Neuron

To begin our study of heterogeneity in biological networks of neurons, we first implemented a reduced model neuron—an endogenous burster that is used in respiratory rhythm generation in the PBC in mammals [8]. Although we are interested in motor pattern-generating systems that are involved in locomotion, we chose this respiratory neuron because it is a simple and generic conductance-based HH model that produces both spiking and bursting. In addition, the reduction in the number of model parameters does not result in a corresponding

degradation in the neural oscillatory dynamics. We used both analog/silicon technologies and digital/computer technologies to implement this reduced model neuron for both single-neuron and population studies. The following sections describe the mathematical equations that were used to implement the model for both the neuron and the synaptic connections.

2.1.1 Neuron Model

A computational model of this reduced model neuron [8] is based on a single-compartment HH formalism [55], and its dynamics are described completely by a set of autonomous differential equations. The transmembrane current, I_{c}, is defined as

$$I_{\mathrm{c}} = C_{\mathrm{mem}} \frac{dV_{\mathrm{mem}}}{dt} = -(I_{\mathrm{Leak}} + I_{\mathrm{NaP}} + I_{\mathrm{Na}} + I_{\mathrm{K}} + I_{\mathrm{syn}}) + I_{\mathrm{app}} \tag{1}$$

where C_{mem} is the whole cell capacitance (21 pF), V_{mem} is the membrane potential, and t is time. The intrinsic currents include a passive leakage current (I_{Leak}), a persistent Na^+ current with slow inactivation (I_{NaP}), a fast Na^+ current (I_{Na}), and a delayed-rectifier K^+ current (I_{K}). The subthreshold currents are I_{Leak} and I_{NaP}, and the spiking currents are I_{Na} and I_{K}. The extrinsic currents comprise an externally applied current (I_{app}) and the sum of the synaptic currents (I_{syn}) from the other N neurons. Note that a calcium concentration variable is not required for the generation of bursting behavior for this model, and this endogenously bursting neuron does not require external stimulation.[1] Figure 1 shows the electrical equivalent circuit of the reduced model neuron.

Voltage-dependent activation and inactivation variables regulate the conductances of the ionic currents (I_{NaP}, I_{Na}, I_{K}), and the dynamics of a single gating variable, x, is given by

$$\frac{dx}{dt} = (x_\infty - x)/\tau_x, \tag{2}$$

$$x_\infty = \left(1 + \exp^{(\frac{V_{\mathrm{mem}} - \theta_x}{\sigma_x})}\right)^{-1}, \text{ and} \tag{3}$$

$$\tau_x = \bar{\tau}_x / \cosh\left(\frac{V_{\mathrm{mem}} - \theta_x}{2\sigma_x}\right). \tag{4}$$

A sigmoidal, steady-state, voltage-dependent (in)activation function of x with a slope that is inversely proportional to σ_x (also referred to as a slope voltage) and a half-(in)activation

[1] Hodgkin and Huxley showed that only Na^+ and K^+ are required for action potential generation [55].

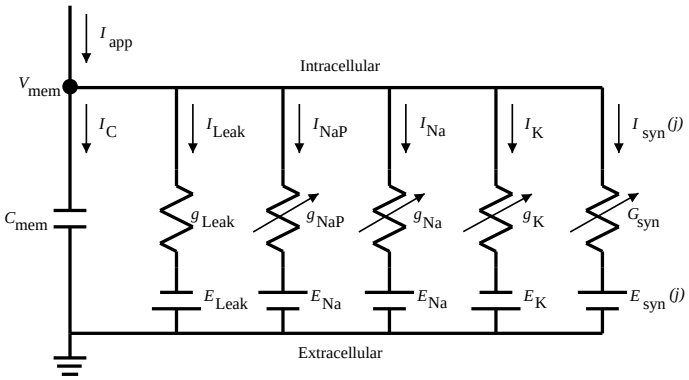


Figure 1: Reduced model neuron—electrical equivalent circuit. The variable resistances represent voltage-dependent conductances. Only one of the N synaptic currents is shown.

at $V_{\text{mem}} = \theta_x$ (also referred to as a half maximal voltage) is given by x_∞. A bell-shaped voltage-dependent time constant that has a maximal value $\bar{\tau}_x$ at $V_{\text{mem}} = \theta_x$ and a half-width determined by σ_x is given by τ_x.

The intrinsic currents, I_i for $i \in \{\text{Leak}, \text{NaP}, \text{Na}, \text{K}\}$, are defined as

$$I_{\text{Leak}} = \bar{g}_{\text{Leak}} \cdot (V_{\text{mem}} - E_{\text{Leak}}) \tag{5}$$

$$I_{\text{NaP}} = \bar{g}_{\text{NaP}} \cdot m_\infty \cdot h \cdot (V_{\text{mem}} - E_{\text{Na}}) \tag{6}$$

$$I_{\text{Na}} = \bar{g}_{\text{Na}} \cdot q_\infty^3 \cdot (1 - n) \cdot (V_{\text{mem}} - E_{\text{Na}}) \tag{7}$$

$$I_{\text{K}} = \bar{g}_{\text{K}} \cdot n^4 \cdot (V_{\text{mem}} - E_{\text{K}}) \tag{8}$$

where $\bar{g}_i$ is the maximal conductance and E_i is the reversal potential. Four gating variables are required (h, n, m_∞, q_∞), and two of them (m_∞, q_∞) activate instantaneously. Note that I_{K} does not have an inactivation term, and the activation of I_{K} and the inactivation of I_{Na} use the same gating variable, n. Table 2 specifies the published, or canonical, parameters of the reduced model neuron [8]. The following should be noted from the table: (1) The model has a fast time constant (10 ms) and a slow time constant (10 s), which represent two state variables (n and h, respectively). V_{mem} represents a third state variable[2]; (2) Activation

[2]A state variable is given by any $\frac{d}{dt}$ term and provides *memory* for the system.

Table 2: Canonical reduced model neuron parameters separated by type—intrinsic currents and gating variables.

Intrinsic Currents		
i	E_i (mV)	$\bar{g}_i$ (nS)
Leak	-65	2.8
NaP	n/a	2.8
Na	50	28.0
K^+	-85	11.2

Gating Variables			
x	θ_x (mV)	σ_x (mV)	$\bar{\tau}_x$ (ms)
h	-48	6	10,000
n	-29	-4	10
m_∞	-40	-6	n/a
q_∞	-34	-5	n/a

(inactivation) is represented by $\sigma_x < 0$ ($\sigma_x > 0$); (3) The two instantaneous activation time constants and the two activation exponents are not considered as model parameters.

2.1.2 Synaptic Connections

To implement a network of neurons, we must define the synaptic equations and the network topology. The synaptic current *to* neuron j, $I_{\text{syn}}(j)$, *from* the population of N neurons is defined as

$$I_{\text{syn}}(j) = \sum_{i=1}^{N} (\bar{g}_{\text{syn}}(i,j) \cdot s(i)) \cdot (V_{\text{mem}}(j) - E_{\text{syn}}(j)) \tag{9}$$

where $\bar{g}_{\text{syn}}(i,j)$ is the maximal synaptic conductance, $s(i)$ is the synaptic gating variable, and E_{syn} is the synaptic reversal potential. Canonical $\bar{g}_{\text{syn}}(i,j)$ values were not defined; instead, a variety of values were used throughout testing with the FPGA model neuron. An all-to-all connectivity scheme in the HCO was implemented—all ipsilateral neurons made excitatory connections ($E_{\text{syn}}(j) = 0$ mV) and all contralateral neurons made inhibitory connections ($E_{\text{syn}}(j) = -80$ mV). The contralateral inhibitory connections provided reciprocal inhibition in the network. Figure 2 shows the synaptic connections required for a four-neuron HCO ($N = 4$). In general, each neuron in an N-neuron configuration will *make* N synaptic connections and *receive* N synaptic connections, resulting in a total of N^2 connections in the network. Note that interneurons were not used.

The dynamics of a single synaptic gating variable *from* neuron i, $s(i)$, are defined by the following equations [9]:

$$\frac{ds(i)}{dt} = \begin{cases} [1 - (1 + k_r) \cdot s(i)]/\tau_s & V_{\text{mem}}(i) > \theta_s \quad \text{(growth rate assumes } s_\infty(i) = 1\text{)} \\ [-k_r \cdot s(i)]/\tau_s & V_{\text{mem}}(i) \leq \theta_s \quad \text{(decay rate assumes } s_\infty(i) = 0\text{)} \end{cases} \tag{10}$$

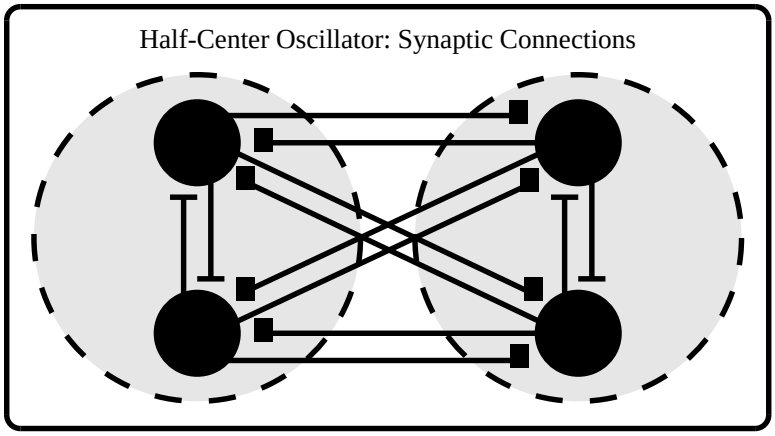


Figure 2: Reduced model neuron—HCO synaptic connections implemented with the FPGA for a four-neuron HCO ($N = 4$). Each half of the HCO is given by a large lightly shaded circle, and the four neurons are given by the large dark circles. The ipsilateral excitatory connections are shown by the small bars, and the contralateral inhibitory connections are shown by the small dark circles.

where $k_r = 1$, $\tau_s = 5$ ms, and $\theta_s = -10$ mV are fixed for all synapses. These equations were simplified from their original form by making the noted assumptions about $s_\infty(i)$. Also note that $s(i)$ is a piecewise function of $V_{\text{mem}}(i)$ and has N values.

These equations for the synaptic connections complete the mathematical description of the reduced model neuron that was used throughout our research. The synaptic connections, however, were only implemented with the digital design, not the analog design, which is the topic of the next section.

2.2 Analog Implementation

Physical systems provide platforms for developing real-world and complex engineered systems. The systems that are inspired by neuromorphic engineering can be used as neural modeling tools that give immediate feedback and intuition to the experimenter because of their real-time nature, which also allows these systems to be interfaced to real systems. Our physical system was implemented with a silicon model neuron, which used a building-block architecture from MOSFET transistors. This latest implementation was motivated by the classical design of Mahowald and Douglas [76] [24], and its architecture was well documented by Simoni [106]. To study network heterogeneity with the silicon model neuron, we also

designed a heterogeneity circuit [96] that will also be discussed in this section.

2.2.1 Architecture and Design Decisions

The high-level functional design of a single conductance is shown in Figure 3. This general building-block architecture facilitates the implementation of a variety of conductance-based neuron models, and we will demonstrate this architectural versatility in Section 3.2 in which we discuss the implementation of three HH models. The steady-state (in)activation sigmoid functions were implemented with differential pair circuits. A current mode [38] lowpass filter circuit [53] generated each (in)activation state variable that could implement a range of time constants, from milliseconds to seconds, but this block was not required for the instantaneous activation variables (m_∞ and q_∞). The output of a current-mode multiplier circuit, normalized by the maximal conductance, was used to bias the output VOTA [19]. The ionic currents were summed at the neuron output node, $V_{\rm mem}$, and this voltage across the membrane capacitance, $C_{\rm mem}$, represented the accumulation of charge on the neural membrane. This current summation is proportional to the derivative of the neuronal membrane potential with respect to time.

The original HH model [55] is more elaborate than the equations that we implemented in silicon; for example, rate-constant equations are employed in the original model. To optimize flexibility and decrease complexity, we made the following architectural design decisions that resulted in approximations to the published reduced model neuron equations: (1) The slopes of the activation and inactivation sigmoidal functions were not controlled externally. Instead, we physically fixed the slopes to one of two constant values using source degeneration (shallower slope) or not (steeper slope)—the $I_{\rm K}$ block used source degeneration and the $I_{\rm NaP}$ and $I_{\rm Na}$ blocks did not. (2) Activation and inactivation powers (*e.g.*, q_∞^3 or n^4) were not implemented, although a fixed approximate exponentiation term, equivalent to the subthreshold transistor parameter κ, where $\kappa < 1$, was generated by the multiplier circuit. From Eq. (3), an exponentiation term has the effect of changing the effective half-activation voltage (*i.e.*, $\theta_x \rightarrow \theta'_x$), the effective sigmoidal slope (*i.e.*, $\sigma_x \rightarrow \sigma'_x$), and the shape of the sigmoid for extreme values (*i.e.*, for $|(V_{\rm mem} - \theta_x)| >> \sigma_x$). (3) The time constants were

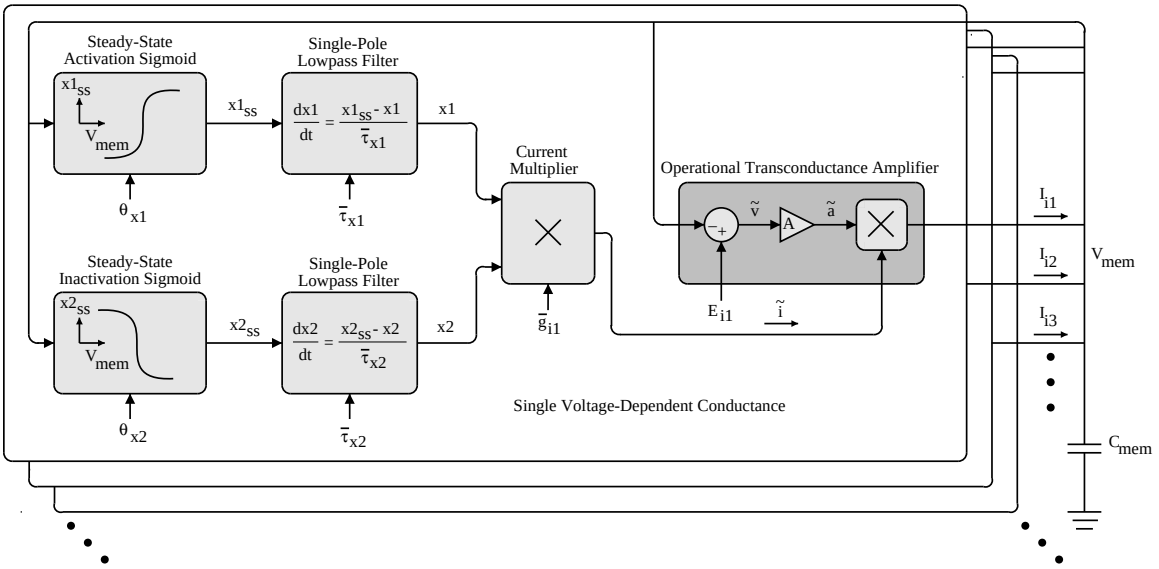


Figure 3: Silicon model neuron building-block architecture—a single voltage-dependent conductance. The ionic currents are given by I_{i1}, I_{i2}, and I_{i3}. Off-chip voltage biases will be used to set the model parameters θ_{x1}, θ_{x2}, $\bar{\tau}_{x1}$, $\bar{\tau}_{x2}$, $\bar{g}_{i1}$, and E_{i1}. An intermediate voltage, current, and gain are represented by $\tilde{v}$, $\tilde{i}$, and $\tilde{a}$, respectively. The state variable $x1$ and its steady-state value $x1_{ss}$, where $x1_{ss} = x1_{\infty}$, are currents (similarly for $x2$ and $x2_{ss}$).

externally set to constant values and were not functions of the membrane potential. Using controllable, but fixed, time constants is a special case of the HH formalism. (4) Strict adherence to a building-block VLSI design precludes coupling between the ionic currents. For example, the published reduced model neuron equations incorporate a reduction in the number of state variables by coupling the I_{K} activation term and the I_{Na} inactivation term, but these state variables were uncoupled in the silicon model neuron (*i.e.*, the $(1-n)$ term in Eq. (7) effectively became a new state variable, p). In addition, all of the reversal potentials were independently controlled externally and were not coupled by ion type, and therefore, E_{Na} and E_{NaP} were separate parameters. These approximations adversely affected the performance of the silicon model neuron, and the negative roles of (2) and (3) will be described in Section 2.2.5. This completes the descriptions of the critical design decisions that we made, and the next section discusses the actual implementation.

2.2.2 Implementation Details

To begin our study of an aVLSI network of heterogeneous neurons, we fabricated an array of 20 nominally identical silicon model neurons that follow the architecture and design decision discussed in the previous section. Figure 4 shows a photomicrograph of the integrated circuit that was manufactured through the MOSIS fabrication service. The circuit design was completed on a Sun Microsystems SunBlade 1000 running the Solaris Unix operating system using the Cadence suite of VLSI CAD tools. The chip was fabricated using a TSMC 0.35 μm process (feature size of 0.2 μm) and was placed in a 40-pin package. The chip was implemented on a die with an area of 2000 μm $\times$ 2000 μm, which included the neuron array and scanner (29% of the area), the pad frame (33%), and the pad frame interconnect area and empty space (38%). This was the first time that we fabricated an array of HH neurons on a single integrated circuit and the first time that we used the 3.3-V TSMC process for our silicon model neurons.

The high-level chip layout is shown in Figure 5. Note that each neuron in the array is isolated because we did not implement synapses with this version of the chip. Heterogeneity circuits are also not included. The tiling scheme required a single off-chip voltage bias to control each parameter across the array because we did not use a floating gate [23] [46] to control each parameter in each neuron. Using the scanner, we were able to select and take off chip the real-time output from only one neuron at a time. We had direct access to one neural output (labeled $V_{\mathrm{mem}}(1)$ in the figure), which allowed us to characterize the voltage-to-physiological relationships for each of the parameters of that neuron (*e.g.*, chip voltage to physiological time constant, chip voltage to physiological conductance, and chip voltage to physiological voltage). These characterizations were considered when we set the parameters of the other neurons. To determine these relationships, we performed a series of physiological voltage clamp experiments [52]. In fact, our ability to replicate voltage-clamp data provided us with an indication of the accuracy of our silicon model neuron. The extracted physiological values, however, were not known with the same level of precision as the physiological units used with a digital implementation, and this uncertainty caused another deviation between the silicon model neuron and the theoretical reduced

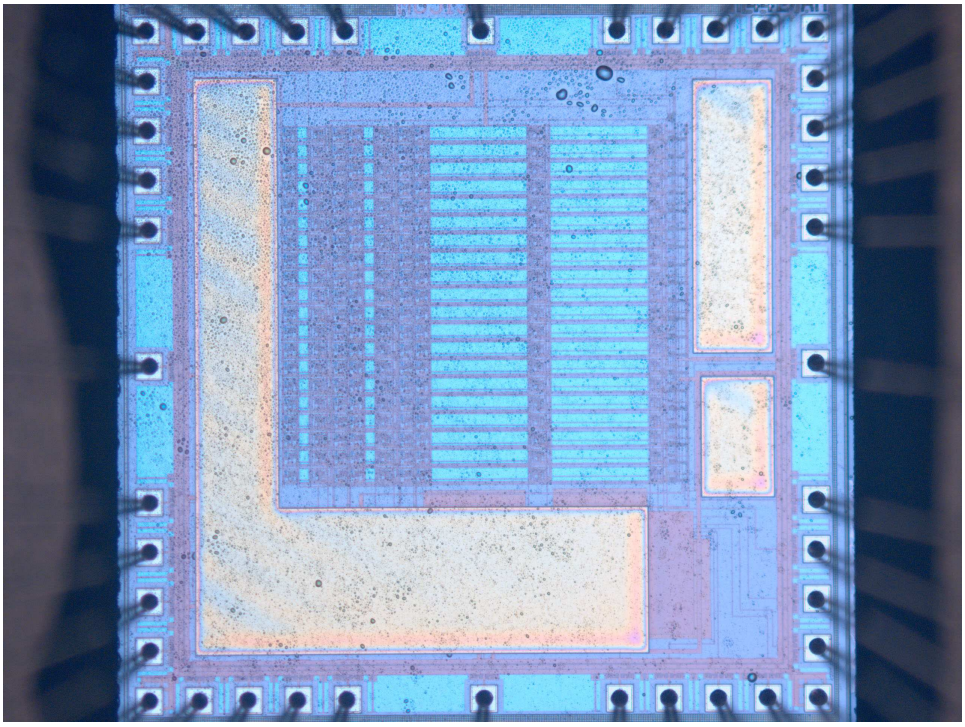


Figure 4: Silicon model neuron implementation—photomicrograph of the array of silicon model neurons.

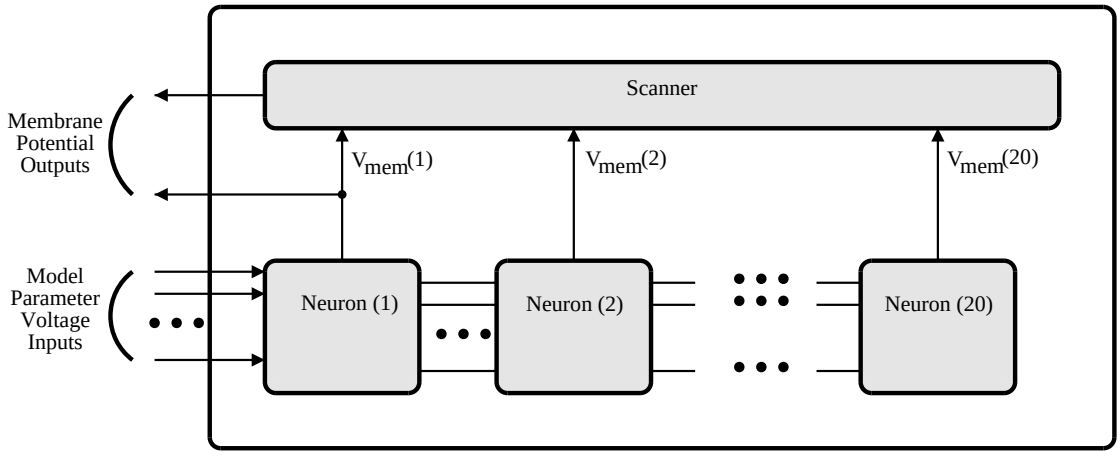


Figure 5: Silicon model neuron implementation—chip layout of 20 isolated neurons plus scanner circuitry.

model neuron. In addition, the resulting voltage-to-physiological relationships are nonlinear (*i.e.*, a change in the voltage bias controlling a maximal conductance does not result in a proportional conductance change).

Table 3 shows the 40 pin definitions of the silicon model neuron. The different functions of the pins include parameter settings (21), voltage biases (5), scanner inputs/outputs (7), a neuron output (1), and those that are not used (6). As explained previously (Section 2.2.1), because of the building-block VLSI design, coupling between the ionic currents was not done. As a result, the n state variable that is coupled in the published equations is de-coupled here. Similarly, two E_{Na} parameters are required for both I_{Na} and I_{NaP}. Only the output of neuron 1 was taken off chip directly because 19 more pins were not available for the other neuron outputs using standard MOSIS packaging. Instead, a scanner was used to take all 20 of the neuron outputs off chip. Because we had access to neuron 1, we referred to it as the instrumented neuron because it was used to determine the voltage-to-physiological mappings for each parameter. Note that none of the state variables were taken off chip.

The first step to use the scanner was to calibrate it to determine its voltage input/output mappings. To do so, we set the scanner's V_{sweepON} pin to V_{dd}, swept an input voltage on the V_{sweep} pin, and read the voltage output on the V_{scan} pin for a few different above-threshold

Table 3: Silicon model neuron implementation—pin definitions.

Number	Name	Type
1	not used	n/a
2	not used	n/a
3	not used	n/a
4	not used	n/a
5	V_{bias1}	bias
6	not used	n/a
7	$V_{\tau n}$	scanner
8	$V_{\tau p}$	scanner
9	V_{sweep}	scanner
10	V_{scan}	scanner
11	V_{sweepON}	scanner
12	$\bar{\tau}_n(I_{\text{K}})$	parameter
13	$\bar{g}_{\text{K}}$	parameter
14	$\sigma_n(I_{\text{K}})$	parameter
15	V_{gnd}	bias
16	$\theta_n(I_{\text{K}})$	parameter
17	V_{bias2}	bias
18	E_{K}	parameter
19	θ_q	parameter
20	σ_q	parameter

Number	Name	Type
21	$\bar{g}_{\text{Na}}$	parameter
22	$\theta_n(I_{\text{Na}})$	parameter
23	$\sigma_n(I_{\text{Na}})$	parameter
24	$\bar{\tau}_n(I_{\text{Na}})$	parameter
25	V_{dd}	bias
26	$E_{\text{Na}}(I_{\text{Na}})$	parameter
27	θ_m	parameter
28	σ_m	parameter
29	$\bar{g}_{\text{NaP}}$	parameter
30	θ_h	parameter
31	σ_h	parameter
32	$\bar{\tau}_h$	parameter
33	$E_{\text{Na}}(I_{\text{NaP}})$	parameter
34	$\bar{g}_{\text{Leak}}$	parameter
35	V_{dd}	bias
36	not used	n/a
37	E_{Leak}	parameter
38	V_{mem} (neuron 1)	output
39	V_{DataIn}	scanner
40	V_{clk}	scanner

$V_{\tau n}$ and $V_{\tau p}$ biases, which control the scanning speed. The larger the current through these bias transistors, the faster the response of the circuit, but the smaller its operating range. Note that this calibration process only sent a voltage through an extra scanner circuit that is located at the end of the scanner circuitry. Therefore, transistor mismatches between this scanner block and the individual scanners for each neuron output created offsets in the calibration data; although the offsets for each of these neuron outputs were not quantified, we were confident that these offsets would not adversely affect the results. We set the $V_{\tau n}$ and $V_{\tau p}$ biases by setting their currents approximately equal to each other by using the relationships [115]

$$I_n \approx \kappa_n (V_{\tau n} - V_{Tn})^2 \tag{11}$$

$$I_p \approx \frac{\kappa_n}{3} (V_{\tau p} - V_{Tp})^2 \tag{12}$$

where V_{Tn} and V_{Tp} are the threshold voltages, which were obtained from the MOSIS website for the particular fabrication run. Note that $V_{\tau p}$ and V_{Tp} are relative to V_{dd} in these equations. The factor of $1/3$ in the equation for I_p is due to the difference in mobility between n-channel and p-channel transistors. For a given $V_{\tau n}$, we solved for $V_{\tau p}$ by assuming that $I_n \approx I_p$:

$$V_{\tau p} = \sqrt{3(V_{\tau n} - V_{Tn})^2} + V_{Tp}. \tag{13}$$

Figure 6 shows the results of the scanner calibration for a 3.3-V TSMC process. As expected, the larger $V_{\tau n}$, the smaller the operating range. From these results, we decided to complete our testing using $V_{\tau n} = 0.9$ V and $V_{\tau p} = 1.94$ V (the absolute value, not relative to V_{dd}). The other scanner inputs, V_{DataIn} and V_{clk}, were then used to read each neuron output; the timing diagrams of these scanner inputs are shown in Figure 7.

2.2.3 Test Setup

Our test setup is shown in Figure 8. Parameter values of the silicon neuron were set in software using MATLAB and then converted to physical chip voltages using 16-bit, 11-V range (-3 V to 8 V) digital-to-analog converters providing 168 μV of resolution per step (built in-house). MATLAB was also used to analyze the neuron output data and required

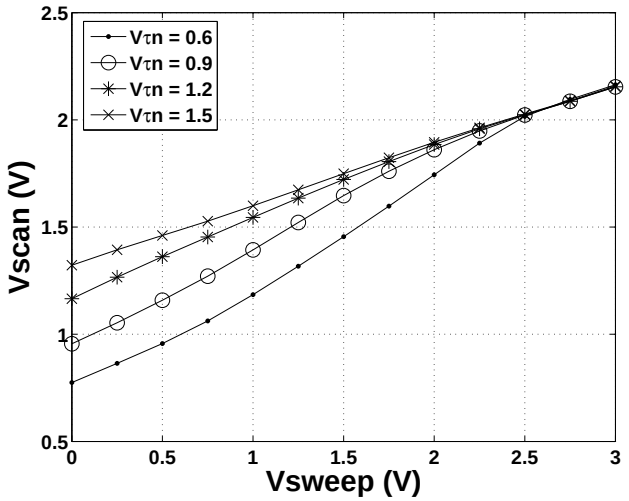


Figure 6: Silicon model neuron implementation—scanner calibration. $V_{\tau n} = 0.6, 0.9, 1.2,$ and 1.5 V, and $V_{\tau p}$ was set according to Eq. (13) for $V_{\rm dd} = 3.3$ V, $V_{Tn} = 0.55$ V, and $V_{Tp} = 0.76$ V (obtained from the MOSIS website).

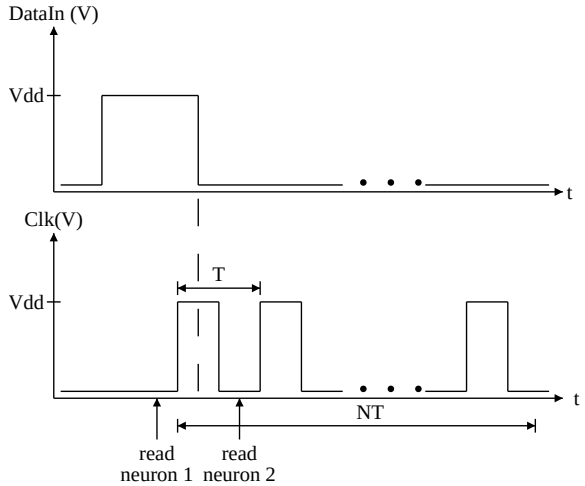


Figure 7: Silicon model neuron implementation—scanner timing diagrams for the $V_{\rm DataIn}$ and $V_{\rm clk}$ signals. To read the next neuron output, the scanner uses a falling edge triggered clock.

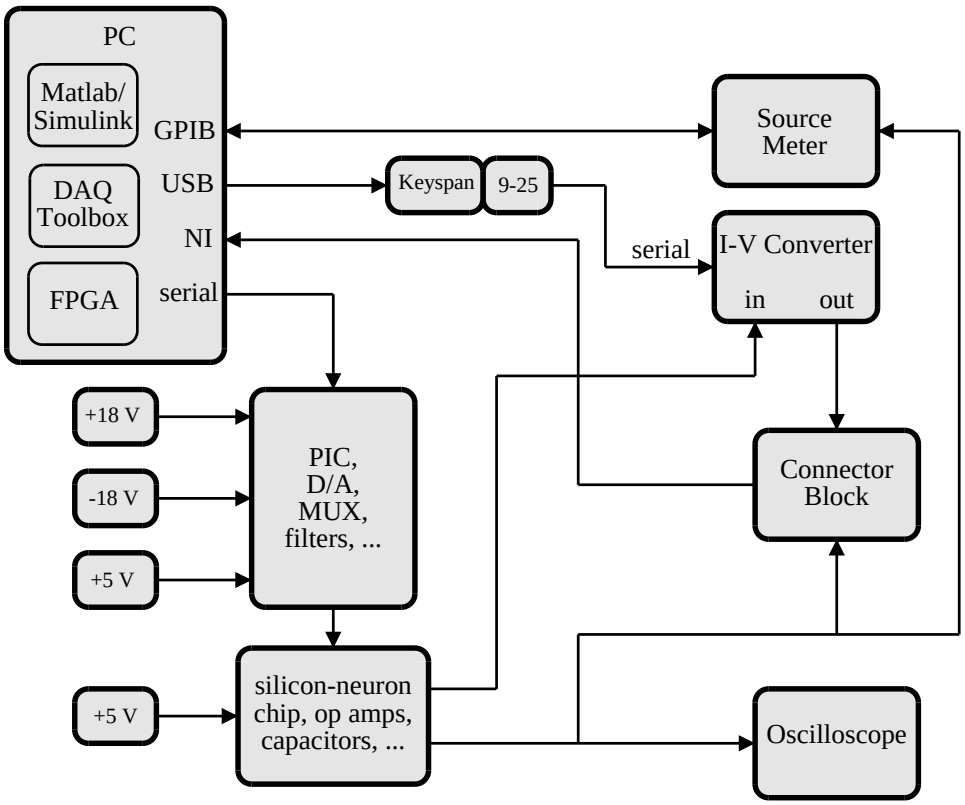


Figure 8: Silicon model neuron implementation—test setup.

the data acquisition (DAQ) toolbox. To collect the data, MATLAB was integrated with a National Instruments PCI-6035E 16-bit data acquisition card in a Dell Dimension 4300 personal computer with a 1.6 GHz Pentium 4 processor and 1 GB of RAM running the Windows 2000 Professional operating system. The conductance characterization tests were performed with a Stanford Research Systems SR570 Low-Noise Current Pre-Amplifier (I-to-V converter), and a Keithley 2400-C SourceMeter was used to do current-step tests. A Fluke 196C portable oscilloscope, a National Instruments BNC-2110 shielded connector block, and power supplies were also required.

The versatile silicon-neuron architecture fabricated on a single integrated-circuit chip (Figure 9) can be viewed as a multi-purpose analog computing machine that accepts some set of inputs (*i.e.*, the HH-model parameters) to obtain a dynamic output (*i.e.*, the neuronal membrane potential, V_{mem}). In addition, software and hardware interfaces were used to control off-chip voltage biases to set the HH-model parameters and to receive and analyze

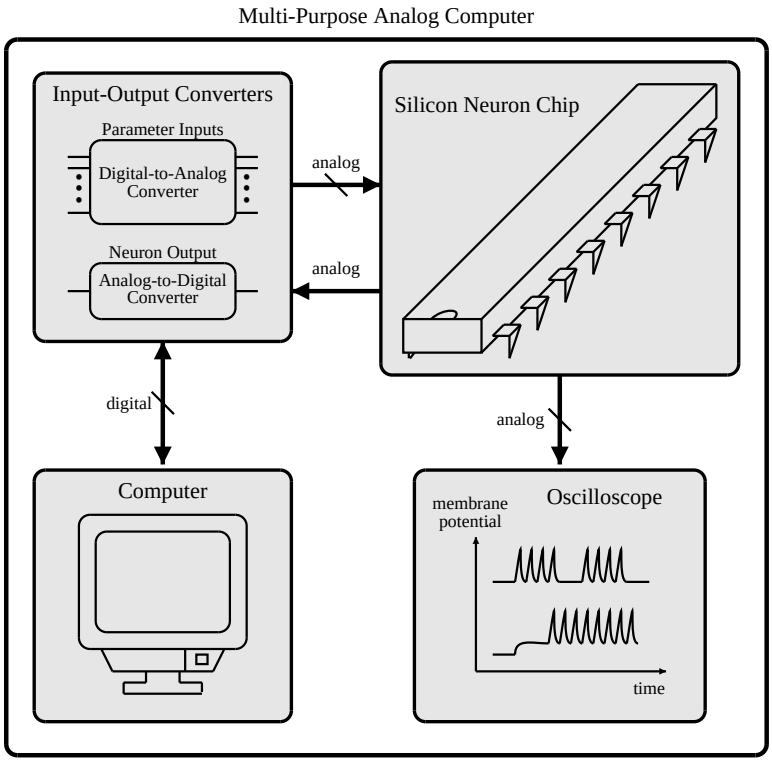


Figure 9: Comparative analysis—the silicon neuron as an analog computing device.

the neural output.

2.2.4 Heterogeneity Circuit

To generate and maintain heterogeneity[3] in our silicon model neuron, we built a heterogeneity circuit [96]—a subthreshold CMOS array whose spatial distribution of output currents approximates a normal distribution [126], which has an independently controllable mean and standard deviation. This Gaussian distribution results naturally from the transistor physics of subthreshold CMOS equations and the symmetry imposed in the network. We demonstrated the usefulness of the circuit array in a system-level example by using the currents to drive integrate-and-fire silicon neurons in which the neuronal input current and resulting interspike interval are inversely related. The circuit and silicon-neuron arrays were fabricated on the same integrated-circuit chip using the MOSIS AMI 0.5 μm process, and

[3]Note that in this context, the term *heterogeneity* can be replaced by *variability*.

testing yielded experimental results that matched well with theoretical analysis. By controlling and measuring the effects of the neurons due to the applied heterogeneity, we planned to investigate how this heterogeneity affects the bursting ranges of each of the neurons. As a result, this heterogeneity circuit was expected to be a building block that could be used to analyze applied heterogeneity that could lead to controllable heterogeneous networks of silicon model neurons.

Previous work has described the generation of arbitrary functions whose mathematical basis is derived from the underlying transistor physics. For example, a CMOS current-mode exponential-function circuit exploits the square-law characteristic of MOS transistors in the saturation region to approximate a Taylor series expansion [25] [127]. An example from the fuzzy-logic community includes a digitally programmable CMOS current-mode circuit that generates a set of nonlinear functions derived from the transcharacteristic of an operational transconductance amplifier (OTA) [99]. That nonlinear function generator [99] and our own previous circuit [6] can generate Gaussian transfer functions, but our heterogeneity circuit generates a normal distribution.

The goal of our work was to implement a subthreshold CMOS array that generated a set of currents whose values approximated a Gaussian distribution; subthreshold CMOS circuits [81] [73] have been employed in a wide variety of applications, including OTA [28], winner-take-all [32], and log-domain filtering [104] circuits. The mean and standard deviation of this Gaussian distribution were independently controllable.[4] We accomplished this goal by mapping a uniformly-distributed set of voltages, implemented with a resistive voltage divider, on a circuit array that implemented a hyperbolic sine function. We will show the mathematical relationships that are necessary to justify this mapping, which also includes the formulation of the key result that the transistor physics of subthreshold CMOS equations, coupled with the symmetry imposed in the network, produce a hyperbolic sine function. We will conclude that section with an explanation of the circuit constraints and provide our circuit and system experimental results.

[4] Although we use probabilistic terminology to discuss this circuit, it does not generate a *random* set of current values, but rather a *deterministic* set based on the independently controllable mean and standard deviation voltage biases. A histogram of the resulting current values approximates a Gaussian.

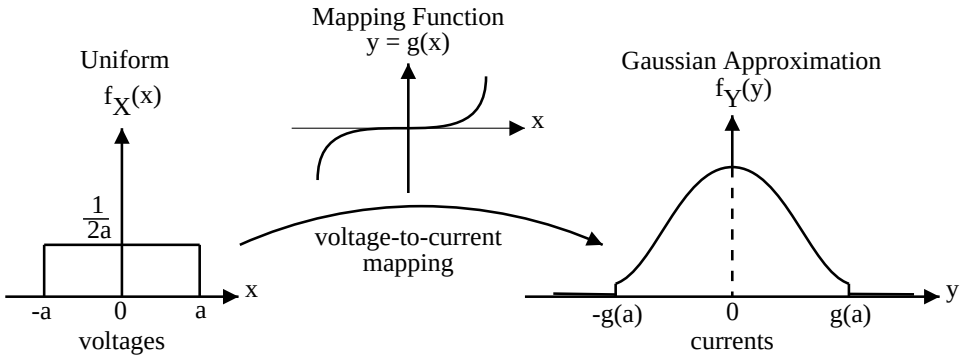


Figure 10: Heterogeneity circuit—mapping a uniform set of voltages ($f_{\mathbf{X}}(x)$) to obtain a distribution of currents ($f_{\mathbf{Y}}(y)$) that approximates a normal distribution. x represents voltages relative to the thermal voltage, and y represents currents. $2a$ is the range of possible voltages.

Mathematical Justification of Mapping Function. The goal of our work was to develop an integrated circuit that implements the mapping function $y = g(x)$ and uses a uniform distribution of voltages, $f_{\mathbf{X}}(x)$, to obtain a distribution of currents, $f_{\mathbf{Y}}(y)$, that approximates a Gaussian distribution as depicted in Figure 10. We obtained the probability density function, $f_{\mathbf{Y}}(y)$, for the special case where $g(x)$ is monotonically increasing, from the following formula [89]:

$$f_{\mathbf{Y}}(y) = \frac{f_{\mathbf{X}}(g^{-1}(y))}{g'(g^{-1}(y))}. \tag{14}$$

From Eq. (14) we determined the following differential equation involving the mapping function:

$$g'(x) = \frac{f_{\mathbf{X}}(x)}{f_{\mathbf{Y}}(g(x))}. \tag{15}$$

To find a solution to Eq. (15), we used the following relationships:

$$F_{\mathbf{X}}(x) = F_{\mathbf{Y}}(y)|_{y=g(x)} \tag{16}$$

$$f_{\mathbf{X}}(x) = \begin{cases} \frac{1}{2a} & -a \leq x \leq a \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

$$F_{\mathbf{Y}}(y) = \frac{1}{2}\Big[1 + \text{erf}\left(\frac{y - \mu_{\mathbf{Y}}}{\sigma_{\mathbf{Y}}\sqrt{2}}\right)\Big] \tag{18}$$

where $F_{\mathbf{Y}}(y)$ and $F_{\mathbf{X}}(x)$ are the cumulative distribution functions for the Gaussian and uniform distributions, respectively. To solve for y in Eq. (18), we used Eq. (16) to replace $F_{\mathbf{Y}}(y)$ with $F_{\mathbf{X}}(x)$ and used Eq. (17) in the relationship $F_{\mathbf{X}}(x) = \int_{-\infty}^{x} f_{\mathbf{X}}(\alpha)d\alpha$. As a result, the implementation of a Gaussian distribution required the following mapping function:

$$y = g(x) = \mu_{\mathbf{Y}} + \sigma_{\mathbf{Y}}\sqrt{2}\,\mathrm{erf}^{-1}\left(\frac{x}{a}\right). \tag{19}$$

The inverse error function is qualitatively matched to the function $A\sinh(Bx)$, which is easier to implement in VLSI circuits. Using a least-squared-error minimization routine to fit the hyperbolic sine function to the inverse error function, we obtained an 18 percent maximum relative error (at the ends of the curves) and an eight percent mean relative error (over the entire curves) between the two functions. See Appendix A for more mathematical details of this circuit. The next section provides the formulation of the hyperbolic sine function.

Circuit Implementation. The circuit shown in Figure 11 is a single element of the subthreshold CMOS array that implements the hyperbolic sine function, the mapping function that is used to generate a Gaussian approximation. The circuit used V_{mean} to set the mean and $V_{\mathrm{max}} - V_{\mathrm{min}}$ to set the standard deviation of the output currents, $i_{\mathrm{out}}(j)$ where $j \in \{1, ..., N\}$, that comprise the Gaussian distribution. V_{mean}, V_{min}, and V_{max} were independently controllable external biases. An expanded view of the voltage divider that sets the gate voltages, $V_n(j)$ and $V_p(j)$, is shown in Figure 12. The voltage divider used nominally identical resistors, and the voltages represent a uniform symmetric distribution such that $V_n(j) = V_p(N - j + 1)$.

Hyperbolic Sine Formulation. The current $i_n(j) - i_p(j)$ in each of the N elements of the array is defined as

$$i_n(j) - i_p(j) = I_o e^{\kappa V_n(j)} - I_o e^{\kappa V_p(j)}, \tag{20}$$

where I_o (pre-exponential subthreshold current) and κ (subthreshold gate-efficiency constant) are the subthreshold process parameters, and all voltages are specified in terms of the thermal voltage [81]. We can explicitly show the symmetrical gate-voltage relationships

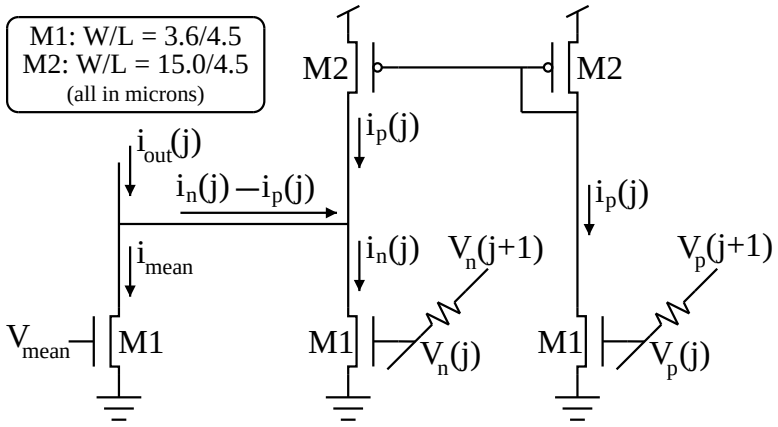


Figure 11: Heterogeneity circuit—element j of the N-element CMOS array that implements the hyperbolic sine function.

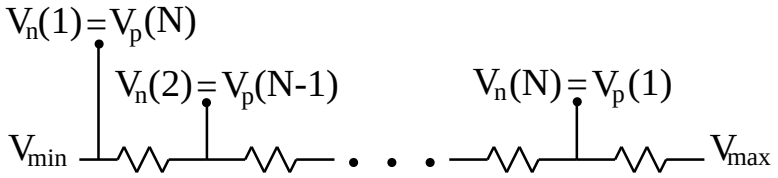


Figure 12: Heterogeneity circuit—the voltage divider that implements the uniform symmetric distribution of gate voltages.

between $V_n(j)$ and $V_p(j)$ by constructing them as

$$V_n(j) = \overline{V}'_{\text{mid}} + V_{\text{diff}}(j) \tag{21a}$$

$$V_p(j) = \overline{V}'_{\text{mid}} - V_{\text{diff}}(j). \tag{21b}$$

These equations use the relationships

$$\overline{V}_{\text{mid}} = \frac{V_{\text{max}} + V_{\text{min}}}{2} \tag{22a}$$

$$\overline{V}_{\text{incr}} = \frac{V_{\text{max}} - V_{\text{min}}}{N} \tag{22b}$$

$$\overline{V}'_{\text{mid}} = \overline{V}_{\text{mid}} - \frac{1}{2}\overline{V}_{\text{incr}} \tag{22c}$$

$$V_{\text{diff}}(j) = \frac{V_n(j) - V_p(j)}{2}. \tag{22d}$$

$\overline{V}_{\text{mid}}$ represents the theoretical mid-voltage of the voltage divider, $\overline{V}_{\text{incr}}$ is the incremental voltage difference between two adjacent nodes in the voltage divider, and $\overline{V}'_{\text{mid}}$ is the adjusted theoretical mid-voltage of the voltage divider.[5] Note that the overbars with $\overline{V}_{\text{mid}}$,

[5]This adjustment is necessary because of the construction of the ends of the voltage divider: $V_{\text{min}} =$

$\overline{V}_{\text{incr}}$, and $\overline{V}'_{\text{mid}}$ indicate constant values for these variables, which are introduced for notational convenience (*i.e.*, they are not physical circuit voltages). In addition, $V_{\text{diff}}(j) = x$ represents a zero-mean, uniform distribution of voltages that generates $f_{\mathbf{X}}(x)$, shown in Figure 10.

Substituting Eq. (21) into Eq. (20) yields the mapping function

$$i_n(j) - i_p(j) = A \sinh(Bx), \tag{23}$$

where $A = 2I_o e^{\kappa \overline{V}'_{\text{mid}}}$ and $B = \kappa$. This hyperbolic sine function is a critical result because its form approximates the Gaussian mapping function given in Eq. (19), and this result also provides another example of how the exponential is a computational primitive [81].

Relationship between Circuit Quantities and Variance. We derived the following relationships from Eq. (23):

$$y = g(x) = A \sinh(Bx) \tag{24a}$$

$$g^{-1}(y) = x = (1/B) \sinh^{-1}(y/A). \tag{24b}$$

To obtain the density function given by Eq. (14), $f_{\mathbf{Y}}(y) = f_{\mathbf{X}}(x)/g'(x)$, we used the following hyperbolic trigonometric functions:

$$\cosh(x) = \frac{1}{2}(e^x + e^{-x}) \tag{25a}$$

$$\sinh^{-1}(x) = \ln(x + \sqrt{x^2 + 1}). \tag{25b}$$

We substituted Eq. (24a) and Eq. (24b) in Eq. (14) where $f_{\mathbf{X}}(x)$ is given in Eq. (17) to obtain the density function implemented with the circuit:

$$f_{\mathbf{Y}}(y) = \frac{1}{2aAB\sqrt{(y/A)^2 + 1}}. \tag{26}$$

We also chose to add a zero-mean, Gaussian distributed random variable $\mathbf{Z}$ to account for transistor mismatch [43], resulting in the stochastic mapping function

$$\mathbf{Y} = \tilde{g}(\mathbf{X}, \mathbf{Z}) = A \sinh(B \cdot \mathbf{X}) + \mathbf{Z} \cdot I_{\text{mean}}, \tag{27}$$

$V_n(1) = V_p(N)$ but $V_{\max} \neq V_n(N) = V_p(1)$.

where $\mathbf{X}$ and $\mathbf{Z}$ are statistically independent and $I_{\text{mean}} = I_o e^{\kappa V_{\text{mean}}}$.

To obtain expressions for the mean ($\mu_{\mathbf{Y}}$) and variance ($\sigma^2_{\mathbf{Y}}$) of the output currents, we can use either probability distribution function, $f_{\mathbf{X}}(x)$ or $f_{\mathbf{Y}}(y)$. For any continuous function $p(\cdot)$, the following general expected value formulas hold:

$$\begin{aligned} \mathrm{E}[p(\mathbf{Y})] &= \int_{-a}^{a} p(g(x)) f_{\mathbf{X}}(x) dx \\ &= \int_{g(-a)}^{g(a)} p(y) f_{\mathbf{Y}}(y) dy. \end{aligned} \tag{28}$$

Using Eq. (27) in Eq. (28) to calculate the variance:

$$\sigma^2_{\mathbf{Y}} = \mathrm{E}[\mathbf{Y}^2] = \mathrm{E}[(A \sinh(B \cdot \mathbf{X}) + \mathbf{Z} \cdot I_{\text{mean}})^2] \tag{29}$$

$$\begin{aligned} &= \int_{-a}^{a} (A \sinh(Bx))^2 f_{\mathbf{X}}(x) dx \\ &\qquad + \int_{-\infty}^{\infty} (z I_{\text{mean}})^2 f_{\mathbf{Z}}(z) dz, \end{aligned} \tag{30}$$

we obtained the following result:

$$\sigma^2_{\mathbf{Y}} = \frac{A^2[\sinh(2aB) - 2aB]}{4aB} + \sigma^2_{\mathbf{Z}} I^2_{\text{mean}}. \tag{31}$$

As expected, $\sigma_{\mathbf{Y}}$ is an increasing function in a with $\sigma_{\mathbf{Y}} \to 0$ as $a \to 0$. Another meaningful result is that $\sigma_{\mathbf{Y}}$ is dependent on $\sigma_{\mathbf{Z}}$ (the standard deviation of $\mathbf{Z}$) but not on $\mu_{\mathbf{Z}}$ (the mean of $\mathbf{Z}$). In addition, $\mu_{\mathbf{Y}} = \mu_{\mathbf{Z}} \cdot I_{\text{mean}} = 0$, because we chose $\mathbf{Z}$ to be zero mean.

Circuit Constraints. To ensure subthreshold operation, we maintained V_{mean} at some margin below the threshold voltage ($V_{\text{th}} \approx 680$ mV for the applicable process and fabrication run), and by definition, $V_{\max} \geq V_{\min}$. Also, $i_p(j) - i_n(j) \leq i_{\text{mean}}$ to guarantee $i_{\text{out}}(j) \geq 0$, which translates into the constraint $V_{\max} \leq V_{\text{mean}}$.[6] As a result, extreme values of the output current were truncated, which made another contribution to the deviation from a true Gaussian distribution. In addition, we required $V_{\max} + 100$ mV $> V_{\text{mean}}$ to produce a standard deviation that is significant relative to the mean.

Experimental Results. We obtained experimental data for a CMOS array ($N = 40$) from a chip fabricated in a MOSIS AMI 0.5 μm process. We determined the mean extracted

[6]Note that the name of the bias voltage $V_{\max}$ was given to indicate that it sets the maximum value of the *range*, not that it is the *maximum* bias voltage.

process parameter values to be $\kappa = 0.60$ and $I_o = 0.7$ fA. To obtain these values, we swept V_{mean} in the subthreshold region between 350 mV and 550 mV in 10-mV increments in each of the 40 elements of the array and recorded the output currents. We then fit the sets of data to exponentials of the form $I = Ae^{BV_{\text{mean}}}$, where $A = I_o$ and $B = \kappa/U_T$. We assumed a thermal voltage, U_T, of 25 mV. We averaged the resulting I_o and κ values to obtain the final extracted values.

Circuit Results. Figure 13 illustrates the experimental normalized standard deviation (normalized to the mean current) for $V_{\text{mean}} = 525$ mV ("+" markers) and for $V_{\text{mean}} = 625$ mV ("×" markers). The two normalized sets of data are approximately equal, indicating that the normalized standard deviations were independent of V_{mean}. A nonzero, persistent normalized standard deviation is also evident for $V_{\max} - V_{\min} < 100$ mV, and it follows this scaling formula:

$$\sigma_1 = \sigma_2 \cdot e^{\kappa(V_1 - V_2)} \tag{32}$$

where σ_1 is the standard deviation for $V_{\text{mean}} = V_1$ (similarly for σ_2). In addition, the theoretical curve for $\sigma_{\mathbf{Z}} = 0$ (solid line) shows a discrepancy with the experimental results that is corrected by using a best-fit to $\sigma_{\mathbf{Z}}$ in the theoretical curve (dashed line), which yielded a result that matches well with the experimental results.

For our application, we required a wide Gaussian current distribution and were not bounded by the locality of the individual biases (*i.e.*, $i_{\text{out}}(j) < i_{\text{out}}(j+1)$ for $j \in \{1, \ldots, N-1\}$ was not a requirement). As a result, we made no effort to decrease the natural transistor offsets, which prevent a narrow distribution, and we simply considered these random offsets as an additive effect to our current distribution. Figure 14 shows the mapping function, $y = g(x)$, which was obtained by plotting the output currents as a function of position along the array. After obtaining the natural offset data (K^+), we subtracted it from the experimental data ($\mathbf{L}$) to obtain the compensated data ($\mathbf{M}$). The compensated data is only included to show the distribution for no offsets, but the experimental data was acceptable for our application. Because these data are related by $\mathbf{M} = \mathbf{L} - \mathrm{K}^+$, their variances are

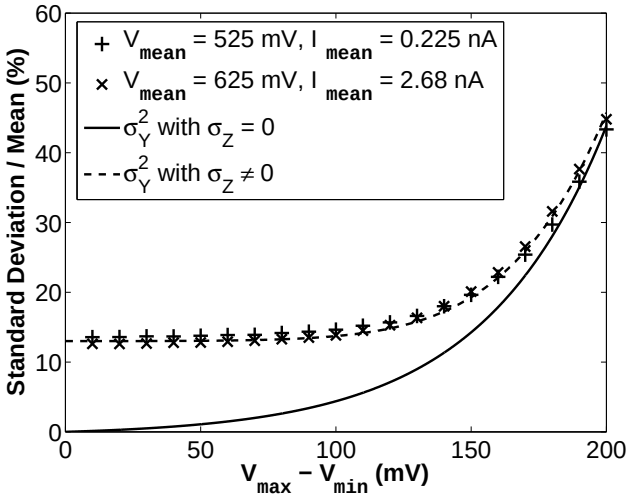


Figure 13: Heterogeneity circuit—experimental and theoretical normalized standard deviations. The lower theoretical curve (solid line) uses $\sigma_{\mathbf{Z}} = 0$, and the upper theoretical curve (dashed line) uses $\sigma_{\mathbf{Z}} = 0.13$. The "+" and "×" markers indicate the experimental values for two different values of V_{mean}.

given by

$$\sigma^2_{\mathbf{M}} = \sigma^2_{\mathbf{L}} + \sigma^2_{\text{K}^+} - 2\sigma_{\mathbf{L}\text{K}^+}. \tag{33}$$

For the data shown in Figure 14, $\sigma^2_{\mathbf{L}} = 0.0408$, $\sigma^2_{\text{K}^+} = 0.0032$ and $\sigma_{\mathbf{L}\text{K}^+} = 0.0054$, resulting in $\sigma^2_{\mathbf{M}} = 0.0332$ (all values are in units of $(\text{nA})^2$). Additionally, we show the theoretical curve Eq. (23) that uses the calculated values $A = 0.12$ nA and $B = 0.6$ based on the mean extracted process parameters, κ and I_o. The variance of the theoretical curve was 0.0325 $(\text{nA})^2$, which is approximately equal to $\sigma^2_{\mathbf{M}}$. Note that $\sigma^2_{\text{K}^+}$ gives the bottom-end limitation of the standard deviation, and $\sigma^2_{\text{K}^+} = 0$ implies perfect transistor matching. If we were required to control a Gaussian distribution smaller than the natural offset distribution, then we would need to reduce the transistor offsets.

System Results. To demonstrate the applicability of the CMOS array, we connected each output current to an input of an integrate-and-fire neuron [35] [36], as depicted in Figure 15. These neurons generated a voltage pulse each time the voltage across the capacitor, C, reached a threshold, and this output voltage was reset immediately after the

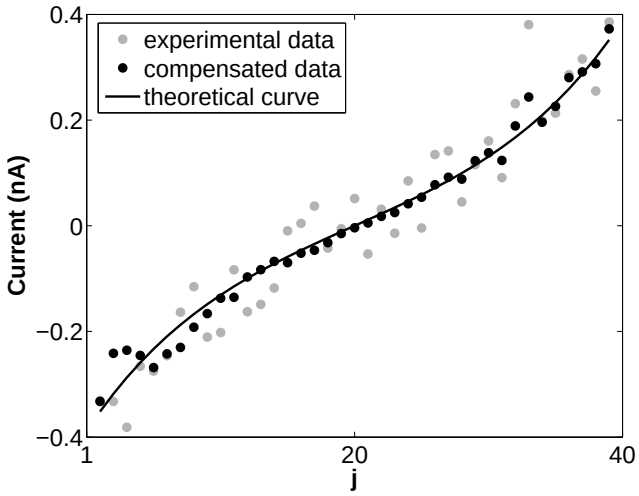


Figure 14: Heterogeneity circuit—the mapping function, $y = g(x)$. The experimental data (light) and compensated data (dark) is shown for $V_{\text{min}} = 400$ mV, $V_{\text{max}} = 550$ mV, and $V_{\text{mean}} = 550$ mV. The solid curve is Eq. (23) using the theoretical values for A and B.

pulse. The magnitude of the input current determined the timing of the pulse, and V_{control}, an independently controllable external bias, set the width of the pulse.

We recorded the mean values of the reciprocals of the interspike intervals for each neuron.[7] Figure 16 shows experimental data in which these mean reciprocals were plotted as a function of position along the subcircuit array. Similarly to our evaluation of the circuit-level data shown in Figure 14, we compensated the output of each neuron (the integrate-and-fire neuron circuits introduce additional mismatch in the circuit) by subtracting the differences from the mean value of the data (with zero applied standard deviation). The A and B parameters from the $A \sinh Bx$ function were unrelated to the integrate-and-fire circuitry. We plotted the true mapping function given in Eq. (19), using the known values for $\mu_{\mathbf{Y}}$ and $\sigma_{\mathbf{Y}}$, alongside the compensated data.

Conclusion. The subthreshold CMOS array generated a set of currents whose values approximated a Gaussian distribution. We verified the theoretical results with experimental data obtained from the circuit implemented in a MOSIS AMI 0.5 μm process. As can be

[7]The reciprocals of the interspike intervals were linearly related to the set of Gaussian-distributed output currents and are therefore used in this analysis.

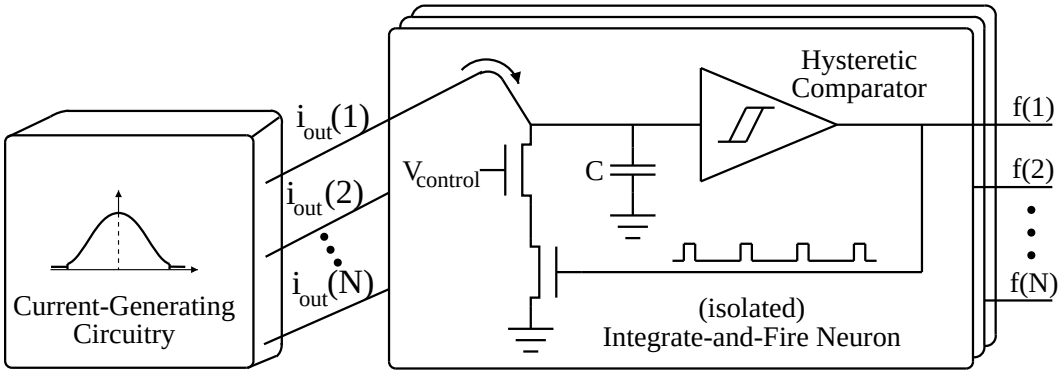


Figure 15: Heterogeneity circuit—system diagram. The input current to neuron j of the N neurons is $i_{\text{out}}(j)$, and $f(j)$ represents the reciprocal of the interspike interval of its output voltage.

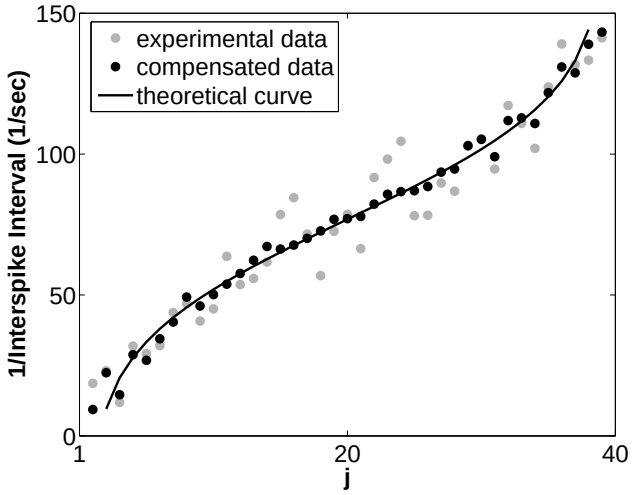


Figure 16: Heterogeneity circuit—system data. Experimental data (light) and compensated data (dark) for $V_{\text{min}} = 300$ mV, $V_{\text{max}} = 450$ mV, and $V_{\text{mean}} = 450$ mV. The solid curve is Eq. (19), the desired mapping, using the experimental values for $\mu_{\mathbf{Y}}$ and $\sigma_{\mathbf{Y}}$.

seen by the experimental data shown in Figure 14 and Figure 16 and because our intended application for the CMOS array would not be affected, we made no effort to improve transistor matching or to add cascoding transistors, which would have reduced the differences between the experimental and theoretical results for this prototype. Also, because VLSI design is particularly applicable for scaling, our 40-element array could have been easily expanded. We later found, however, that generating and controlling heterogeneity and setting the synaptic connections using the HH-based silicon model neuron was too difficult to obtain the biological results that we desired. As a result, we decided to switch to an FPGA platform. The analysis and quantification of the differences between the analog and digital implementations that were used to make our decision are given in the next section.

2.2.5 Analog Approximation Limitations

Although many differences exist between the canonical published equations and the ones implemented in our silicon model neuron (Section 2.2.1), the two most important differences were the following: (1) the published voltage-dependent time-constant equations were implemented as constant values and (2) the exponents of the activation terms were implemented with approximately unity exponents. We analyzed the role of these two differences using a MATLAB-SIMULINK model neuron.

First, recall Eq. (4), the voltage-dependent time-constant equation, which is repeated below:

$$\tau_x = \bar{\tau}_x / \cosh\left(\frac{V_{\text{mem}} - \theta_x}{2\sigma_x}\right) \tag{34}$$

where σ_x is the slope factor. The function flattens as σ_x increases, and Figure 17 shows this dependence. Only two time constants were required in the model: fast (milliseconds) represented by the n state variable and slow (seconds) represented by the h state variable.

We modified Eq. (7) and Eq. (8) in the following way to parameterize the two non-unity activation exponents:

$$I_{\text{Na}} = \bar{g}_{\text{Na}} \cdot q_{\infty}^{a} \cdot (1 - n) \cdot (V_{\text{mem}} - E_{\text{Na}}) \tag{35}$$

$$I_{\text{K}} = \bar{g}_{\text{K}} \cdot n^{b} \cdot (V_{\text{mem}} - E_{\text{K}}) \tag{36}$$

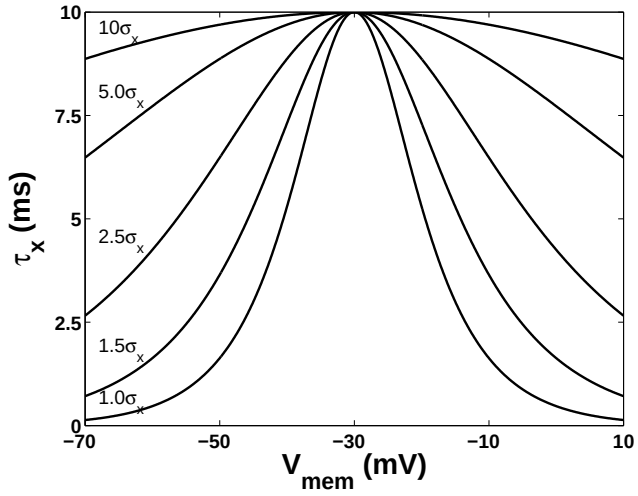
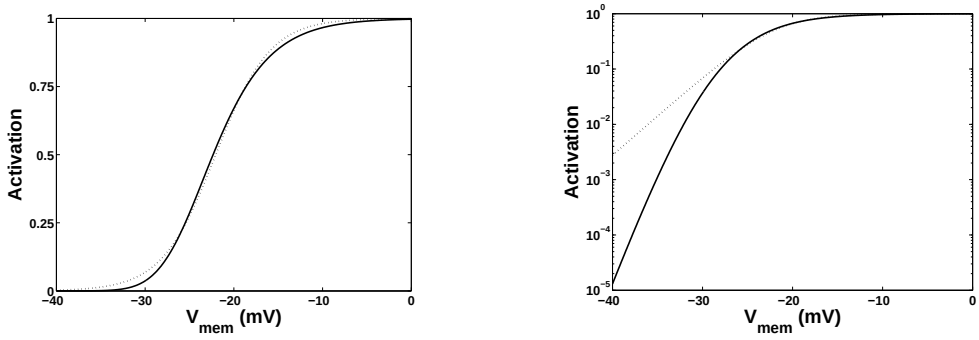


Figure 17: Comparison between the published equations and the silicon model neuron—the role of the voltage-dependent time-constant slope factor. The plots show the transformation of the voltage-dependent time-constant function as the effective slope increases (as given by Eq. (34)) for $\sigma_x = 4$, $\theta_x = -30$, and $\bar{\tau}_x = 10$.

where $a = 3$ and $b = 4$ in the published equations.

We then transformed the exponents by using a constant-increment method in which $a \in \{1.0, 1.5, 2.0, 2.5, 3.0\}$ and $b \in \{1.00, 1.75, 2.50, 3.25, 4.00\}$. As the exponents were changed, new θ_x and σ_x values were found that minimized a least-squared-error difference between m^3 and the new function using m^a for the different values of a (and similarly for n^4, n^b, and b). This algorithmic method was used to preserve the same steady-state activation functions as much as possible and to preclude any arbitrary changes in those model parameters. In addition, we had control over the θ_x and σ_x parameters in the silicon model neuron so these changes were realizable in our physical system. Figure 18 shows an example of this fit. If a perfect fit could have been accomplished, then no differences would be seen as the activation exponents changed; although small, a difference between the curves clearly exists (Figure 18A), and this difference is more pronounced for activation values between 0.0 and 0.1 and is evident when a log scale is used with the y-axis (Figure 18B).

Another effect of the change in the exponentiation terms was a reduction in the time constant because the changes to θ_x and σ_x also affected the time-constant equation given by



A. B.

Figure 18: Comparison between the published equations and the silicon model neuron—finding the best fit to the steady-state activation function as the exponent was changed. The published equation for this curve (solid line) uses $\theta_x = -29.0$ mV, $\sigma_x = -4.0$ mV, and $b = 4$. The least-squared-error curve fit (dashed line) for $b = 1.0$ results in $\theta_x = -22.0$ mV and $\sigma_x = -3.05$ mV. A. Linear y-axis. B. Log y-axis.

Eq. (4). Figure 19 shows the relationship between a sigmoid function with a unity exponent and a non-unity exponent and a normalized voltage-dependent time constant function. In this example, for $b = 1$ the activation value equals 0.5 at $\theta_x = -30$ mV, but $\theta_x \approx -23.3$ mV for $b = 4$. As a result, changes in the θ_x value will shift the voltage-dependent time-constant equation. Consequently, the smaller the exponentiation term, the larger the effective time constant.

Next, we considered the transformations of the time-constant functions, which required more consideration than for that of the exponentiation terms. One idea was to vary $\bar{\tau}_x$ to maintain a constant area under the time-constant curve as the exponentiation term changed. The area under the curve is given by:

$$\text{area} = \int_{\theta_x-50}^{\theta_x+50} \frac{\bar{\tau}_x}{\cosh\left(\frac{V_{\text{mem}}-\theta_x}{2\cdot\sigma_x}\right)}\, dV_{\text{mem}} \tag{37a}$$

$$= \int_{\theta_x-50}^{\theta_x+50} \bar{\tau}_x \cdot \text{sech}\left(\frac{V_{\text{mem}}-\theta_x}{2\cdot\sigma_x}\right)\, dV_{\text{mem}} \tag{37b}$$

$$= 2\cdot\bar{\tau}_x\cdot\sigma_x\cdot\tan^{-1}\left(\sinh\left(\frac{V_{\text{mem}}-\theta_x}{2\cdot\sigma_x}\right)\right)\Big|_{\theta_x-50}^{\theta_x+50} \tag{37c}$$

$$= 4\cdot\bar{\tau}_x\cdot\sigma_x\cdot\tan^{-1}\left(\frac{50}{2\cdot\sigma_x}\right) \tag{37d}$$

where the full range of the limits of integration (100 mV) was chosen to include virtually the entire relevant V_{mem} space. This method, however, only resulted in bursting when the

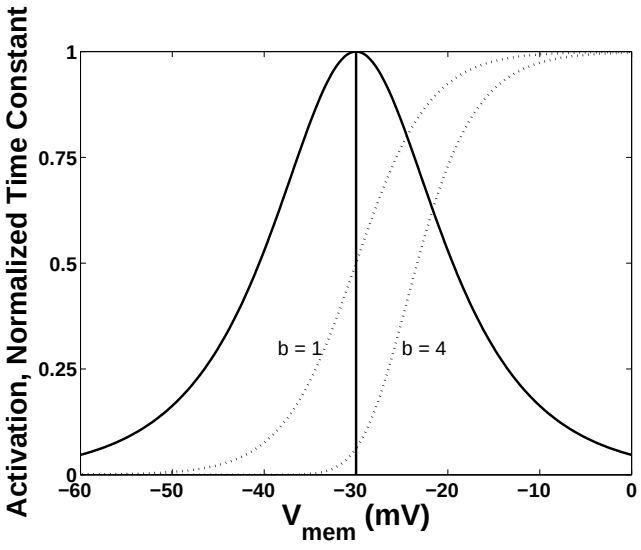


Figure 19: Comparison between the published equations and the silicon model neuron—the effect on the (normalized) time constant as θ_x and σ_x are varied. The normalized time-constant function (solid line) and two activation sigmoid functions (dashed lines) using different exponentiation terms are shown.

published parameters were used, and we determined that the problem was related to the fast time constant. As a result, we used this method for the slow time constant[8], but we used fixed fast time constants; again, this is consistent with our desire to use algorithmic, non-arbitrary methods for changing the model parameters.

Figure 20 shows how the parameterization techniques that we used with the voltage-dependent time-constant function and the activation exponents allowed us to evolve the published equations into the equations that were implemented with the silicon model neuron, resulting in a grid of 25 possibilities. The published (implemented) equations correspond to the upper left (lower right) of the grid, and the top row (first column) correspond to the published exponents (time-constant equations). The parameterizations of the time-constant functions are given in Table 4, and the parameterizations of the activation-exponent functions are given in Table 5. Note that we used a constant value of $\bar{\tau}_x = 7$ ms for all but the first column to obtain adequate bursting regions for the top row of the grid. The differences

[8] An underlying subthreshold oscillation occurred for every case that we considered.

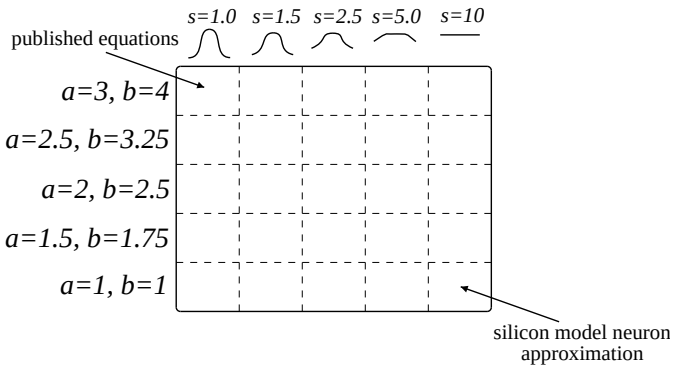


Figure 20: Comparison between the published equations and the silicon model neuron—the transformation of the voltage-dependent time constants and the activation exponents. The upper left grid point represents the published model (full voltage-dependent time constants, published exponents), and the lower right grid point represents the silicon model neuron (flat voltage-dependent time constants, unity exponents). Figure 21 shows the burst mappings in $\bar{g}_{\mathrm{NaP}} - E_{\mathrm{Leak}}$ space for each of the grid points.

Table 4: Comparison between the published equations and the silicon model neuron—the transformation of the time-constant parameters. The first row corresponds to the published values.

	slow (h) $\bar{\tau}_h$ (s)	fast (n) $\bar{\tau}_n$ (ms)
$1.0\sigma_x$	10.0	10.0
$1.5\sigma_x$	7.1	7.0
$2.5\sigma_x$	5.1	7.0
$5.0\sigma_x$	4.1	7.0
$10\sigma_x$	3.8	7.0

between the sigmoidal plots given by the parameters in Table 5 and the canonical sigmoidal plots are all similar to that shown in Figure 18.

Figure 21 shows the burst mapping diagrams as E_{Leak} and $\bar{g}_{\mathrm{NaP}}$ were varied for the cases that corresponded to the grid shown in Figure 20. As expected, as the exponentiation terms were reduced from their published values (looking down each column) and as the voltage-dependent time-constant function flattened (looking across each row), a smaller percentage of the region resulted in bursting activity. For the case in which the exponents were unity and the time constant was fixed (lower right case), all activity (not only bursting) was virtually nonexistent. Bursting only occurred in the top three rows at the published parameter values for E_{Leak} (60 mV) and $\bar{g}_{\mathrm{NaP}}$ (2.8 nS).

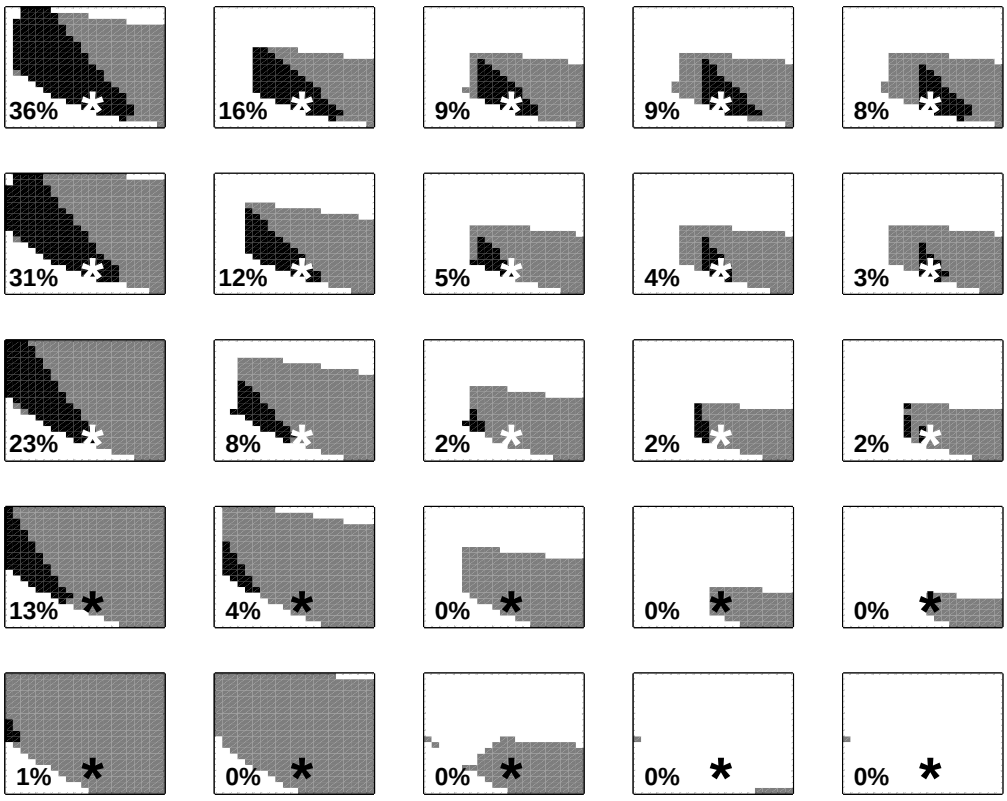


Figure 21: Comparison between the published equations and the silicon model neuron—burst mapping diagrams as E_{Leak} and $\bar{g}_{\text{NaP}}$ were varied for the cases that correspond to the grid shown in Figure 20. The x-axis shows the range $-65 < E_{\text{Leak}} < -55$ mV, and the y-axis shows the range $2 < \bar{g}_{\text{NaP}} < 7$ nS. $E_{\text{Leak}} = -60$ mV and $\bar{g}_{\text{NaP}} = 2.8$ nS are the published parameter values and are indicated by the *. The E_{Leak} increment size is 0.5 mV, and the $\bar{g}_{\text{NaP}}$ increment size is 0.25 nS. The black region is bursting, the white region is silent, and the grey region is neither bursting nor silent (*e.g.*, tonic firing). The number in the bottom left-hand corner of each diagram is the percent of the region that is bursting.

Table 5: Comparison between the published equations and the silicon model neuron—the transformation of the activation-exponent parameters. The first row corresponds to the published values.

Na			K		
a	θ_m	σ_m	b	θ_n	σ_n
3.0	−34.0	−5.00	4.00	−29.0	−4.00
2.5	−32.8	−4.88	3.25	−27.9	−3.91
2.0	−31.3	−4.71	2.50	−26.6	−3.78
1.5	−29.4	−4.44	1.75	−24.8	−3.55
1.0	−26.9	−3.97	1.00	−22.0	−3.05

Figure 22 shows episodes of time for the published values of $E_{\text{Leak}} = -60$ mV and $\bar{g}_{\text{NaP}} = 2.8$ nS as the exponents were transformed to unity values (first column of the grid shown in Figure 20). As the exponents were reduced from their published values, the period of the bursting also reduced. In the ranges of $1.5 < a < 2.0$ and $1.75 < b < 2.5$, the bursting evolved into tonic firing, and the spike frequency of this tonic firing increased as the exponents were further reduced.

Although the curve-fit difference appears to be small in Figure 18, the exponentiation transformation resulted in a virtual complete loss of bursting for the unity exponents (bottom row of the grid) even for the case in which the published voltage-dependent time-constant functions were used (first column of the grid). As a result, the exponentiation term clearly provided an extra degree of freedom in this nonlinear dynamical system without which the differences from the published equations multiplied quickly; otherwise, the authors of these and other HH equations would not have required the additional exponentiation term if the θ_x and σ_x parameters were sufficient to model these steady-state activation functions. Mathematically, the form of the differential equation changed as a result of the use of exponents—using $y = x^n$ and solving for dy/dt gives a much different result than one obtained from the simple first-order differential equation for dx/dt.

A comparison between the original steady-state activation function and its least-squared-errors fit shows that the largest difference occurs for activation values between 0.0 and 0.1 (Figure 18B). Because the activation of I_{Na}, and thus the initiation of action potentials, was dependent on these small activation values, a difference in this part of the steady-state

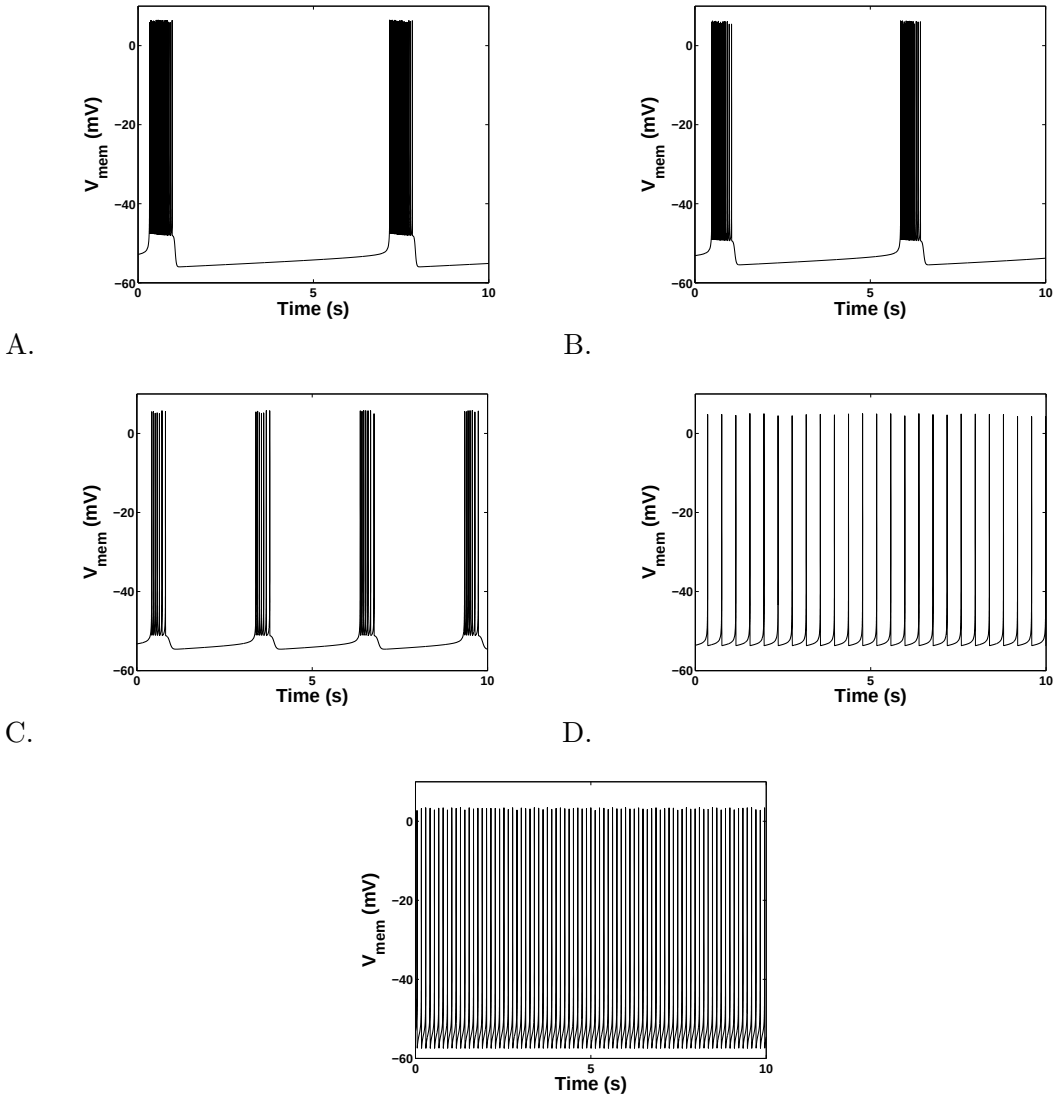


Figure 22: Comparison between the published equations and the silicon model neuron—episodes of time as the exponents were transformed to unity values for $E_{\mathrm{Leak}} = -60$ mV and $\bar{g}_{\mathrm{NaP}} = 2.8$ nS (the first column from Figure 21). A. $a = 3$, $b = 4$. B. $a = 2.5$, $b = 3.25$. C. $a = 2.0$, $b = 2.5$. D. $a = 1.5$, $b = 1.75$. E. $a = 1$, $b = 1$.

activation function may preclude bursting activity for the neuron and consequently may have provided another reason for the virtual loss of bursting as the exponents transformed to unity values. As a result, a modified curve fit could also have been used to minimize these differences for small activation values, but larger differences would have resulted in the other sections of the steady-state activation curve. Finally, the sparse grey regions shown in the upper left hand part of most of the burst mapping diagrams in Figure 21 are artifacts of the increment sizes (0.5 mV for E_{Leak} and 0.25 nS for $\bar{g}_{\text{NaP}}$); for smaller sized increments, a thin continuous grey region would have been shown.

As a result of these findings, we decided that the silicon model neuron was not an optimal platform to continue our study of heterogeneous networks. Instead, we believed that the generation and control of heterogeneity in a network of neurons would be easier and the results would be superior using an FPGA platform. The description of this FPGA implementation is given in the next section.

2.3 Digital Implementation

The digital implementation of the reduced model neuron was constructed using an FPGA. Because it allowed us to easily program and control network heterogeneity, this design proved to be much more flexible than the silicon model neuron and gave us a platform in which scientific results were more easily obtainable. In this section, we discuss the FPGA model neuron implementation and issues such as the network architecture, FPGA resources, and performance.

2.3.1 Implementation of the Matlab-Simulink Model Neuron

We constructed a digital version of the reduced model neuron using MATLAB-SIMULINK that did not differ from the published equations and that was used with our single-neuron studies. We used the results from this MATLAB-SIMULINK model neuron as our reference model. For example, Table 6 shows the individual parameter ranges to maintain bursting when only that single parameter is swept. Note that a random point in parameter space taken from these ranges simultaneously would not likely result in bursting. The following are some of the time-based bursting characteristics resulting from the canonical set of parameters:

Table 6: MATLAB-SIMULINK model neuron—parameter ranges to maintain bursting.

Parameter	Minimum	Canonical	Maximum
Reversal Potentials (mV)			
E_{Leak}	-60.5	-60.0	-56.9
E_{Na}	10	50	1400
E_{K}	-115	-85	-65
Maximal Conductances (nS)			
$\bar{g}_{\text{Leak}}$	1.80	2.80	2.90
$\bar{g}_{\text{NaP}}$	2.70	2.80	5.10
$\bar{g}_{\text{Na}}$	8.0	28	500
$\bar{g}_{\text{K}}$	6.5	11.2	24
Half Maximal Voltages (mV)			
θ_h	-48.8	-48.0	-41.8
θ_n	-37.0	-29.0	-19.0
θ_m	-43.0	-40.0	-39.7
θ_q	-38.0	-34.0	-20.0
Slope Voltages (mV)			
σ_h	4.0	6.0	7.0
σ_n	-8.0	-4.0	-3.0
σ_m	-7.5	-6.0	-5.9
σ_q	-9.5	-5.0	-1.0
Time Constants (ms)			
τ_h	1,500	10,000	22,000
τ_n	4.5	10	16

(1) Period: 6.85 sec. (2) Duty cycle: 9.4%. (3) Starting spike frequency: 59 Hz. (4) Ending spike frequency: 18 Hz. (5) Average spike frequency: 39 Hz. (6) Spikes per burst: 26. The bursting classification algorithm that was used to determine the bursting ranges shown in the table is the same one that will be used throughout the analysis of the FPGA model neuron and is described in the next section. A population study using the MATLAB-SIMULINK model neuron was not feasible because of slow performance; a 30-second simulation of a tonically firing output required 30 seconds of real time *for just a single neuron*. As a result, we switched to an FPGA platform to build a network of neurons.

2.3.2 Classification of Bursting

The bursting-classification algorithm that we used was a critical component in the analysis of the data from the FPGA model neuron. It allowed us to maintain consistency with

comparisons that we made between the homogeneous and heterogeneous configurations. For example, we only considered simple bursting (a single repeatable cycle of relatively constant period and duty cycle consisting of a bursting phase followed by a silent phase) and not complex bursting (a single repeatable cycle consisting of multiple bursting and silent phases with a variety of durations). We also required the bursting amplitudes to be approximately constant. These considerations led to the following quantifiable bursting measures:

- The number of spikes per burst was greater than 3.
- The standard deviation of the peak voltages was less than 4 mV.
- No bursting phase was four times as long as another bursting phase.
- The period was less than 10 seconds.

Although we used a consistent algorithm for classifying bursting, the algorithm itself comprised subjective rules that precluded other types of bursting that would otherwise meet a qualitative definition. As a result, when showing two-dimensional bursting regions, we limited the increment size of the maximal conductances in these plots to either 0.5 nS, 1 nS, or 2 nS because additional information could not be ascertained from smaller increment sizes.

2.3.3 General Architecture of the FPGA Model Neuron

We ported the MATLAB-SIMULINK model neuron to an FPGA using the Xilinx XtremeDSP Development Kit-II, which allowed the FPGA logic blocks to be designed and compiled within the SIMULINK environment.[9] Figure 23 shows the architecture of the FPGA model network that comprised $N = 36$ neurons. This network size resulted in a good tradeoff between analytical simplicity, statistical significance, and implementation complexity. In fact, one study found that at least 30 neurons were necessary to achieve dynamics that scaled to larger populations, thus avoiding "small-number effects" [9]. In addition, with

[9] A large portion of the implementation details of the FPGA model neuron described in the following sections is in a paper by Weinstein, Reid, and Lee that was accepted by IEEE Transactions on Neural Systems and Rehabilitation Engineering [122].

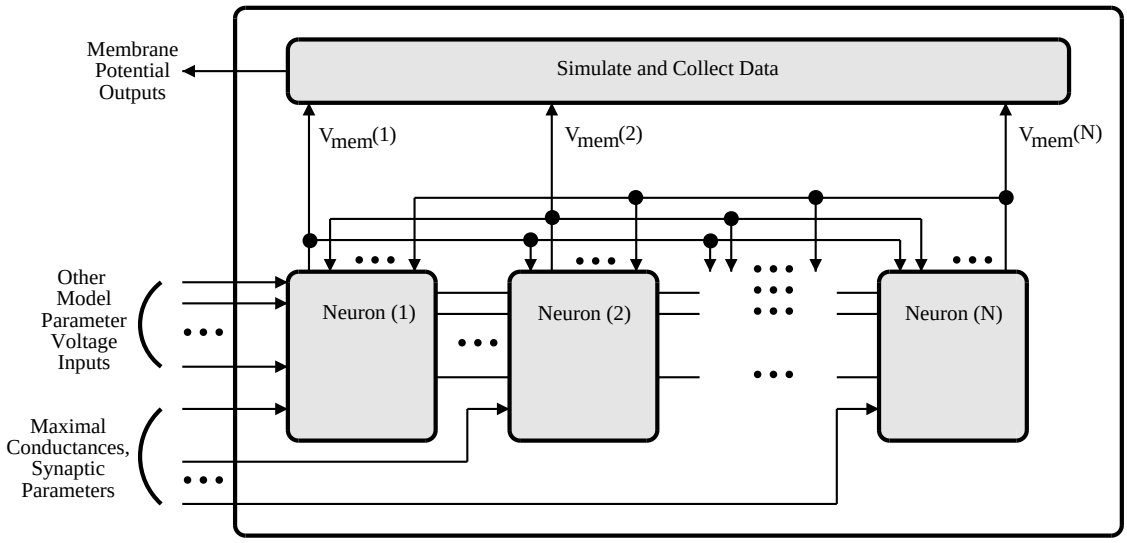


Figure 23: FPGA model neuron—the single FPGA architecture using N neurons to create a HCO. All neurons are connected to each other, but a synaptic weight can be programmed to zero to "break" a connection.

computer simulations, symmetrical effects from the architecture play a role when you have less than that number of neurons.

Because of controllable maximal conductances (including synaptic weights), only one FPGA architecture was required to be implemented, and this flexible design resulted in the maximum number of possible neural connectivity configurations for a given number of neurons—an all-to-all connectivity scheme was implemented resulting in N^2 connections (Figure 2). Other possible connectivity schemes that we considered included a mesh (one half of the HCO would be on top and the other half would be on the bottom), a hypercube, and a small-world network. Note that the N synaptic weights that corresponded to the connections *from* and *to* the same neuron were always set to 0 nS. Because $N^2 >> N$, the number of connections in our network was a much more important factor than the number of neurons. The synaptic weights were independent of the spiking frequency and were not implemented with distant-dependent delays (*i.e.*, long-distance connections were not applicable).

We used a state-driven, pipelined architecture to implement the FPGA model neuron

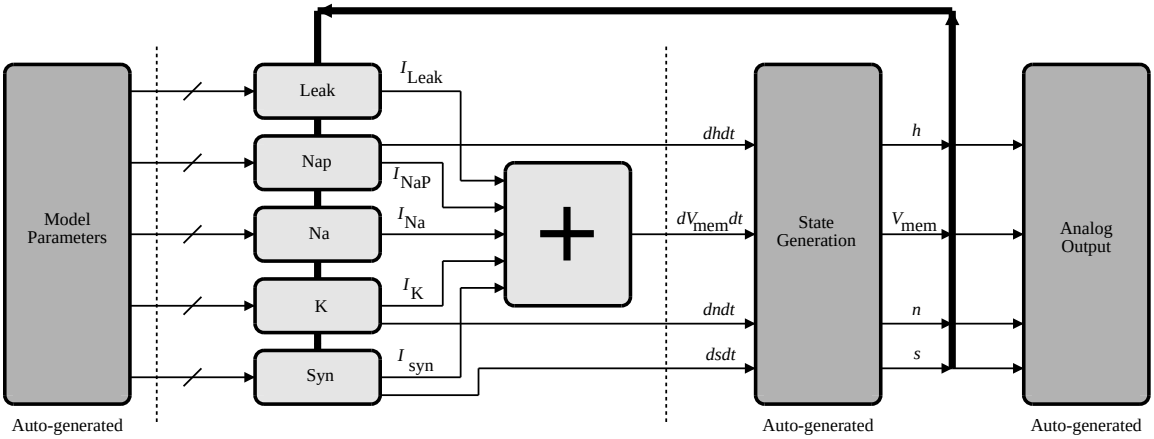


Figure 24: FPGA model network—state-driven, pipelined architecture. The three darkly shaded blocks comprise the auto-generated components—the parameter subsystem, the state subsystem, and the analog output selector. The lightly shaded blocks indicate those systems that are manually generated. The dark feedback line indicates state information propagating back into the data-path.

[121] [122] (Figure 24). Although only one model is shown in the figure, N models are simultaneously simulated through the pipeline. For each of the four state variables (V_{mem}, h, n, s), N pipelined calculations were performed every t_s seconds, where $t_s = 10$ μs is the (approximate) integration time step.[10] The minimum possible N for the network was determined by the latency of the longest algebraic pathway in the pipeline. Figure 25 shows the implementation of the h state variable, where h is defined from Eq. (2), Eq. (3), and Eq. (4). For this example, the longest latency in the pipeline is 15, the algebraic sum of the delays for V_{mem}. The synchronization of these signals throughout the system is critical. Note that only one address needs to be calculated for the two lookup tables, which are required for the nonlinear calculations for h_∞ (shown as hss in the figure) and $\Delta T/\tau_h$ (shown as $\frac{dt}{t_h}$ in the figure).

The hierarchical synaptic network that implements Eq. (9) is shown in Figure 26. For clarity, the figure is only shown for $N = 8$, but the synaptic network is easily scalable. Each $\bar{g}_{\mathrm{syn}}$ parameter contains N values; for example, the second element of $\bar{g}_{\mathrm{syn}}(1)$ is the

[10] A requirement of the system is that t_s/N must be a rational number; therefore, for $N = 36$, t_s was set to 9.9 μs, and $t_s/N = 275$ ns.

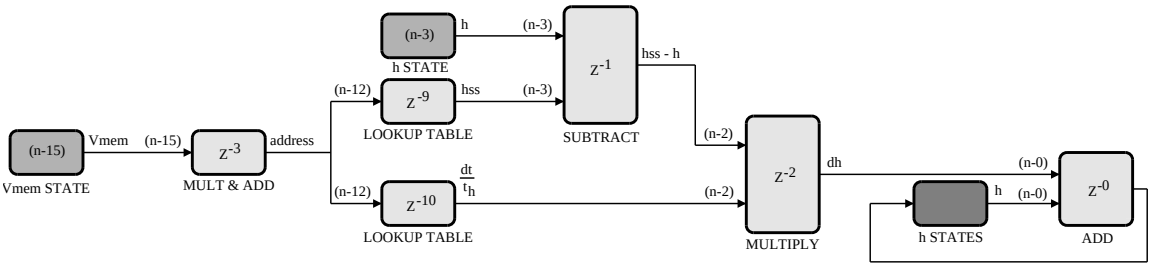


Figure 25: FPGA model neuron—state-driven implementation of the h state variable. The lightly shaded blocks represent the mathematical operations (addition, subtraction, multiplication, and lookup tables), and the pipeline delays are shown in these blocks. The two medium-shaded blocks represent the pipelined state variables, V_{mem} and h. The one darkly-shaded block represents the state-holding register. The numbers in parenthesis represent the pipeline delay of the signal.

weight of the synapse *from* Neuron 1 *to* Neuron 2. Two E_{syn} parameters are required—one implements inhibitory connections and one implements excitatory connections. For an eight-neuron HCO ($N = 8$), Neurons $1-4$ would represent one half of the HCO, and Neurons $5-8$ would represent the other half. One of the N Registers is updated every cycle with a new value of the s state variable and is held constant for N cycles; the counter, relational, register, and constant blocks implement this scheme to update the state variable on the proper line. Note that an encoder block is effectively created from the constant and relational blocks in the dashed region.

2.3.4 FPGA Resources, Signal Precisions, and Quantization Errors

We implemented the FPGA model neuron using the software and hardware listed in Table 7. The maximum and utilized FPGA resources are shown in Table 8 for the $N = 36$ implementation. Note that each slice comprises two flip flops and two lookup tables. The important factors driving the resource utilization were the required precisions of the state variables and the number of neurons, N. Table 9 shows the precisions of both the state variables and the parameters. The minimum precisions of the intrinsic state variables (V_{mem}, h, and n) were determined from their smallest changes (ΔV_{mem}, Δh, and Δn) in the MATLAB-SIMULINK model neuron for a simulation step size of 10^{-5} s. Because we found that $41-43$ bits were required for each of these state variables, we decided to increase all of the precisions to 51 bits because multiple-of-17 precisions results in the most accuracy for

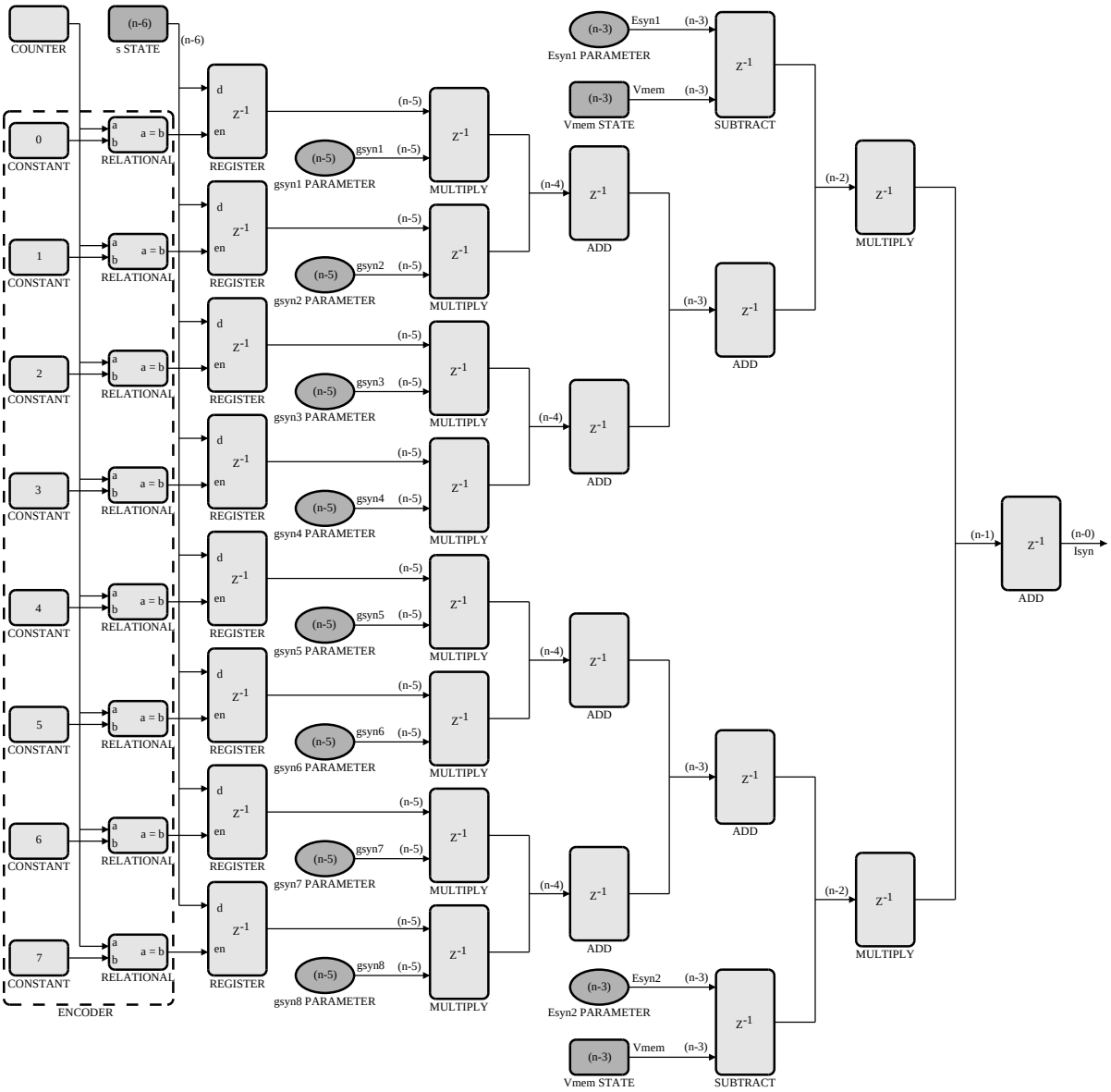


Figure 26: FPGA model neuron—synaptic network for $N = 8$. The lightly shaded blocks represent the mathematical and data operations (addition, subtraction, multiplication, counter, relational, register, and constant), and the pipeline delays are shown in these blocks. The three medium-shaded blocks represent the pipelined state variables, V_{mem} and s. The medium-shaded ovals represent the pipelined variables, $\bar{g}_{\text{syn}}$ and E_{syn}. The numbers in parentheses indicate the pipeline delays of the signals.

Table 7: FPGA model neuron—system specification.

Company	Product
Xilinx (Longmont, CO)	FPGA (Virtex-IV) Xtreme DSP Development Kit IV ISE, System Generator 7.1
Nallatech (Glasgow, UK)	BenONE, BenADDA, DIME-II FUSE 210
The MathWorks (Natick, MA)	MATLAB 7.0.4 SIMULINK 6.2

Table 8: FPGA model neuron—maximum and utilized allowable resources.

Resource	Maximum	Utilized
Slices	15360	15358 (99%)
Flip Flops	30720	21632 (70%)
Lookup Tables	30720	22890 (74%)
Embedded Multipliers/DSP48s	192	127 (66%)
Block RAMs/FIFO 16s	192	179 (93%)

a given level of resources. For example, the number of embedded multipliers used in a multiplication operation is given by the following:

$$\text{number of embedded multipliers} = \lceil \frac{\text{bits}_{\text{input1}}}{17} \rceil \times \lceil \frac{\text{bits}_{\text{input2}}}{17} \rceil \tag{38}$$

where $\lceil c \rceil$ gives the smallest integer not less than c (*i.e.*, the ceiling function). For example, if $\text{bits}_{\text{input1}} = 13$ and $\text{bits}_{\text{input2}} = 28$, then two embedded multipliers would be required, which is the same number that would be required for $\text{bits}_{\text{input1}} = 17$ and $\text{bits}_{\text{input2}} = 34$, which gives more accuracy. Note that this is not a completely free result—although no additional multipliers would be used in this example, the number of adders used to sum the partial products would increase because of the additional bits and the performance would also be slightly affected.

The precision of the synaptic state variables (s) was determined after considering a number of factors such as Eq. (38), quantization error, the target size of the network, the number of FPGA multiplier resources, and the System Generator place-and-route program. N multiplier blocks were required for a network size of N to calculate the N products of $\bar{g}_{\text{syn}}$ and s. Because the precision of $\bar{g}_{\text{syn}}$ was less than 17 bits, the precision of s would

Table 9: FPGA model neuron—state-variable and parameter precisions. The initial conditions of the state variables are shown in parentheses and could not be changed programmatically.

	Qty	Type	Total Bits	Frac Bits	Min	Max	Resolution
State Variables							
V_{mem} (-55 mV)	N	signed	51	43	-128	$+128$	$\approx 10^{-13}$
h (0.5)	N	unsigned	51	51	0.0	1.0	$\approx 4 \times 10^{-16}$
n (0.5)	N	unsigned	51	51	0.0	1.0	$\approx 4 \times 10^{-16}$
s (0.1)	N	unsigned	34	34	0.0	1.0	$\approx 4 \times 10^{-16}$
Parameters							
E_{Leak} (-60 mV)	N	signed	16	8	-128	$+128$	$\approx 4 \times 10^{-3}$
$\bar{g}_{\text{Leak}}$ (2.8 nS)	N	unsigned	16	9	0	128	$\approx 2 \times 10^{-3}$
$\bar{g}_{\text{NaP}}$ (2.8 nS)	N	unsigned	16	9	0	128	$\approx 2 \times 10^{-3}$
$\bar{g}_{\text{Na}}$ (28 nS)	N	unsigned	16	9	0	128	$\approx 2 \times 10^{-3}$
$\bar{g}_{\text{K}}$ (11.2 nS)	N	unsigned	16	9	0	128	$\approx 2 \times 10^{-3}$
$\bar{g}_{\text{syn}}$ (0 nS)	N^2	unsigned	15	9	0	64	$\approx 2 \times 10^{-3}$

determine whether one, two, or three FPGA multiplier resources were required for each of the N multiplication operations. For example, for 51-bit precision for s, $N \times 3$ FPGA multipliers would be required. Because we designed for $N = 40$, 120 multipliers would be required for these synaptic multiplications, resulting in a total of 167 FPGA multipliers in the design, 25 multipliers below the maximum. The design, however, failed to compile because of a combination of resource utilization and the System Generator place-and-route program. As a result, we limited the precision of s to 34 bits, resulting in a reduction of 40 multipliers. Because the calculation of s did not require a lookup table for a nonlinear calculation, it was much simpler than the calculations of the other three state variables. As a result, we felt that the smaller precision of s would not adversely affect the calculations from quantization error.

Six nonlinear calculations from the intrinsic model required lookup tables[11]—four sigmoidal functions for the steady-state activation and inactivation values (m_∞, h_∞, q_∞, and n_∞) and two hyperbolic cosine functions for the time constant values (τ_h and τ_n). The

[11]Note that these lookup tables, with specific precisions and table sizes that match inputs to outputs, have a different meaning from the Lookup Tables that are included in the FPGA resources in Table 8.

number of block RAMs used by one lookup table is given by the following:

$$\text{number of block RAMs} = \lceil \frac{\text{bits}_{\text{output}}}{18} \rceil \times 2^{\text{address\_lines}-10}. \tag{39}$$

As a good tradeoff between quantization error and resource usage, we implemented these tables with 2^{14} entries (14 address lines) and used 18-bit precision with each output. Therefore, we used 2^4 block RAMs for each of the six Lookup Tables for a total of 96 block RAMs. In addition, some of the intrinsic model variables were designated as parameters within the FPGA (*e.g.*, E_{Leak}, $\bar{g}_{\text{Leak}}$, $\bar{g}_{\text{NaP}}$, $\bar{g}_{\text{Na}}$, and $\bar{g}_{\text{K}}$), requiring a single block RAM for each, and the N^2 maximal synaptic conductances ($\bar{g}_{\text{syn}}(i,j)$) required N block RAMs. Finally, each V_{mem} output was sent to a first-in, first-out (FIFO) block, which required an additional N block RAMs.

To determine the inputs of the Lookup Tables, the following quantities were first defined:

$$\phi_x(V_{\text{mem}}) = \frac{V_{\text{mem}} - \theta_x}{\sigma_x} \text{ (for the sigmoidal functions)} \tag{40a}$$

$$\phi_x(V_{\text{mem}}) = \frac{V_{\text{mem}} - \theta_x}{2\sigma_x} \text{ (for the hyperbolic cosine functions)} \tag{40b}$$

where Eq. (40a) is the argument of the exponential function in Eq. (3) and Eq. (40b) is the argument of the hyperbolic cosine function in Eq. (4). The range of V_{mem} for all Lookup Tables was defined between -70 mV (address 0) and $+30$ mV (address $2^{14} - 1$). The address, as a linear function of V_{mem}, is given by the following:

$$\text{address} = (2^{14} - 1)(\frac{V_{\text{mem}}}{100} + 0.7) + 0.5. \tag{41}$$

The additional offset of 0.5 was required because the quantization flag of the adder was set to `truncate`, and the overflow flag was set to `saturate`. Note that only one address calculation was necessary because all of the six Lookup Tables required the same address, which saved resources.

Table 10 shows the first and last elements of the arrays of inputs required for each of the Lookup Tables, and the magnitudes of the step sizes of each of the arrays are given by:

$$|\text{step size}| = \frac{|\phi_x(-70)| + |\phi_x(+30)|}{2^{14} - 1}. \tag{42}$$

Table 10: FPGA model neuron—first and last elements of the lookup tables.

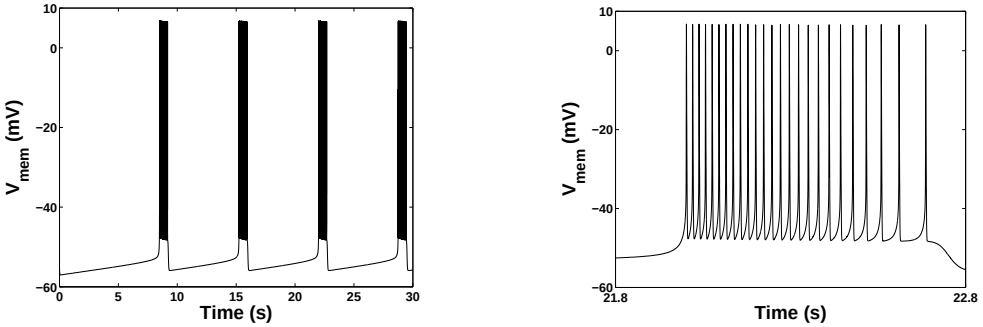
	$\phi_x(-70)$	$\phi_x(+30)$
Sigmoidal Functions		
m_∞	5.0	$-11.\overline{6}$
h_∞	$-3.\overline{6}$	13.0
q_∞	7.2	-12.8
n_∞	10.25	-14.75
Hyperbolic Cosine Functions		
τ_h	$-1.8\overline{3}$	6.5
τ_n	5.125	-7.375

Table 11: FPGA model neuron—mean quantization errors for different lookup table sizes.

Number of Lookup Table Inputs	Mean Quantization Error
2^{10}	$20e^{-5}$
2^{11}	$10e^{-5}$
2^{12}	$6e^{-5}$
2^{13}	$3e^{-5}$
2^{14}	$2e^{-5}$

In addition, to increase the accuracy of the Lookup Tables for τ_h, the output was first multiplied by 2^{11}, to place the maximum value of the table in the 0.5 to 1.0 range, and later an 11-bit shift operation was used to adjust the result back to the correct value.

We also used MATLAB to analyze the quantization error for different sizes of the LUTs. We randomly picked 10^5 values of V_{mem} for $-70 \leq V_{\text{mem}} \leq +30$ mV and compared the actual value of the sigmoidal or hyperbolic cosine function to the quantized FPGA value and recorded the mean errors for each of the LUTs in Table 11. As expected, a doubling of the size of the lookup table roughly corresponded to a fifty percent decrease in the mean quantization error in the table. This analysis led us to the 0.5 additive offset factor that is used in Eq. (41) after we compared the results from testing three different values in 0.5 increments (0.0, 0.5, and 1.0).



A. B.

Figure 27: Comparison between the MATLAB-SIMULINK model neuron and the FPGA model neuron—single-neuron bursting using the canonical parameter values. A. 30-second simulation. B. One bursting phase from A.

2.3.5 Comparison between the Matlab-Simulink Model Neuron and the FPGA Model Neuron: A Further Examination of the Quantization Error

To validate the FPGA model neuron, we performed a comparison between the SIMULINK and FPGA versions of the model for an isolated neuron. We compared the fixed-point single-neuron bursting characteristics of the FPGA model neuron to its floating-point SIMULINK counterpart for sweeps of $\bar{g}_{\text{NaP}}$, $\bar{g}_{\text{Leak}}$, and E_{Leak} using the bursting classification algorithm described in Section 2.3.2. Figure 27A shows a 30-second simulation of the MATLAB-SIMULINK model neuron using the canonical parameters, and Figure 27B shows a close up of one of the bursting phases. The SIMULINK simulation was configured with a variable-step, ode23s (stiff/Mod. Rosenbrock) solver with a relative tolerance of 10^{-6}.

Figure 28 and Figure 31 show how the bursting characteristics vary between the two versions of the model for the sweeps of the three intrinsic parameters. For these figures, we used increment sizes of $\bar{g}_{\text{NaP}} = 0.1$ nS, $\bar{g}_{\text{Leak}} = 0.1$ nS, and $E_{\text{Leak}} = 0.2$ mV. Figure 28 shows the periods (upper data) and the spiking times (lower data). Note that the silent times and duty cycles can be found implicitly from this data. The spiking time is approximately constant for all three sweeps, but a clear correlation between period (and thus duty cycle) and the three intrinsic parameters can be seen. The canonical values of the intrinsic parameters also clearly exist toward the edges of the bursting ranges. Compared to bursting ranges for the SIMULINK model neuron, the FPGA model neuron did not burst for the full range of

Table 12: Comparison between the MATLAB-SIMULINK model neuron and the FPGA model neuron—bursting ranges for three intrinsic parameters.

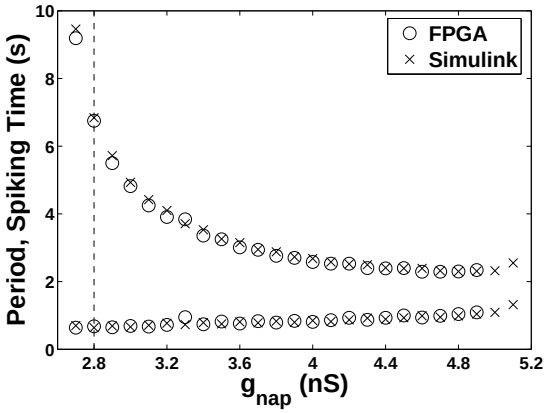
Parameter	Increment	MATLAB-SIMULINK Model Neuron	FPGA Model Neuron
$\bar{g}_{\text{NaP}}$	0.1 nS	$2.7 - 5.1$ nS	$2.8 - 4.9$ nS
$\bar{g}_{\text{Leak}}$	0.1 nS	$1.8 - 2.9$ nS	$1.8 - 2.9$ nS
E_{Leak}	0.2 mV	$-60.4 - -57.0$ mV	$-60.2 - -57.2$ mV

$\bar{g}_{\text{NaP}}$ and E_{Leak}, but did so for $\bar{g}_{\text{Leak}}$ (Table 12). This slightly smaller bursting range is one manifestation of the quantization error introduced in the FPGA model neuron. Another one can be seen at the $\bar{g}_{\text{NaP}} = 3.3$ nS point in which the data from the FPGA model neuron does not follow the data as closely as the other points. Both of these quantization errors could be reduced if we increased the precisions of both the lookup tables and signals[12] throughout the model, but this would have required more FPGA resources, resulting in a smaller network. We felt that the level of quantization error shown in these figures was acceptable for the allowable network size.

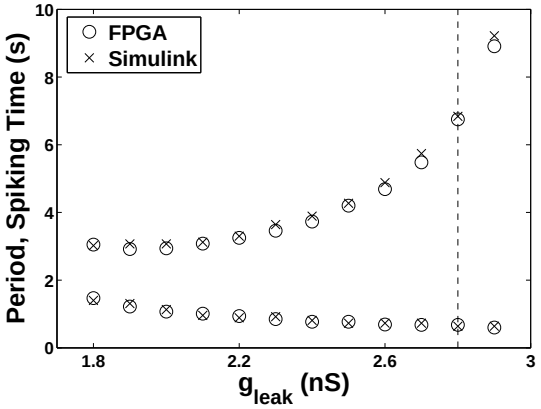
To examine the quantization error further, we repeated Figure 28A (for $\bar{g}_{\text{NaP}}$ only) but reduced the resolution of $\bar{g}_{\text{NaP}}$ by a factor of four from 0.100 nS to 0.025 nS (Figure 29). This data was repeatable. Although the neuron model is a complex nonlinear dynamical system, the quantization errors from the periods and spiking times (time-based quantities) are clearly deterministic, and this result is more evident in Figure 30 in which the differences between the FPGA data and the SIMULINK data are shown. For example, the data for $\bar{g}_{\text{NaP}} = 3.3$ nS is an outlier in Figure 28A but is clearly following patterns in Figure 29 and Figure 30.

We examined additional bursting characteristics (spikes per burst and average spike frequencies) for the two models for sweeps of the three intrinsic parameters (Figure 31). An approximately linear relationship exists between the spikes per burst and these three parameters. A correlation between average spike frequency and $\bar{g}_{\text{Leak}}$ and E_{Leak} is also evident. For the $\bar{g}_{\text{NaP}}$ sweep, however, the average spike frequency is approximately constant,

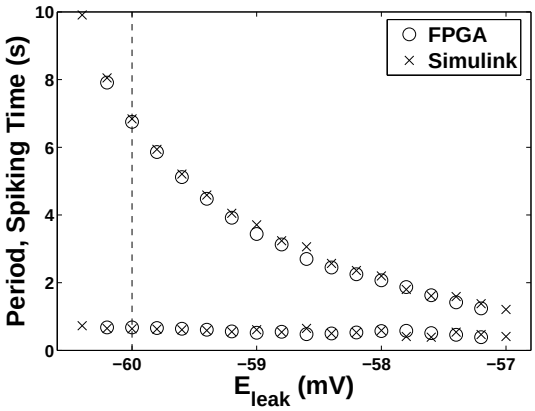
[12]Note that the quantization errors due to the precisions of the model parameters only represent a portion of this signal precision. For example, $\bar{g}_{\text{NaP}} = 3.3$ nS is represented as 3.298828125 nS when only nine fractional bits are used.



A.

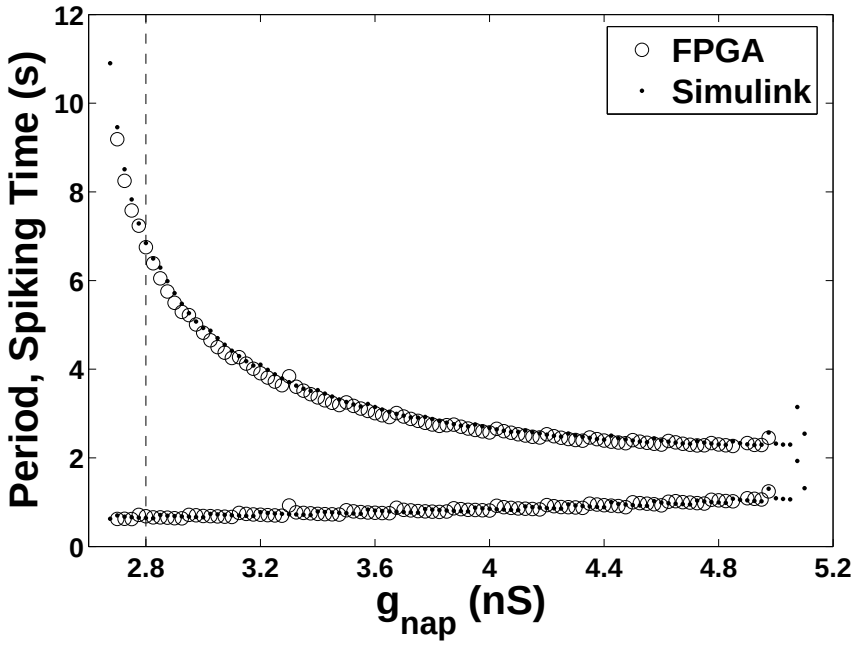


B.

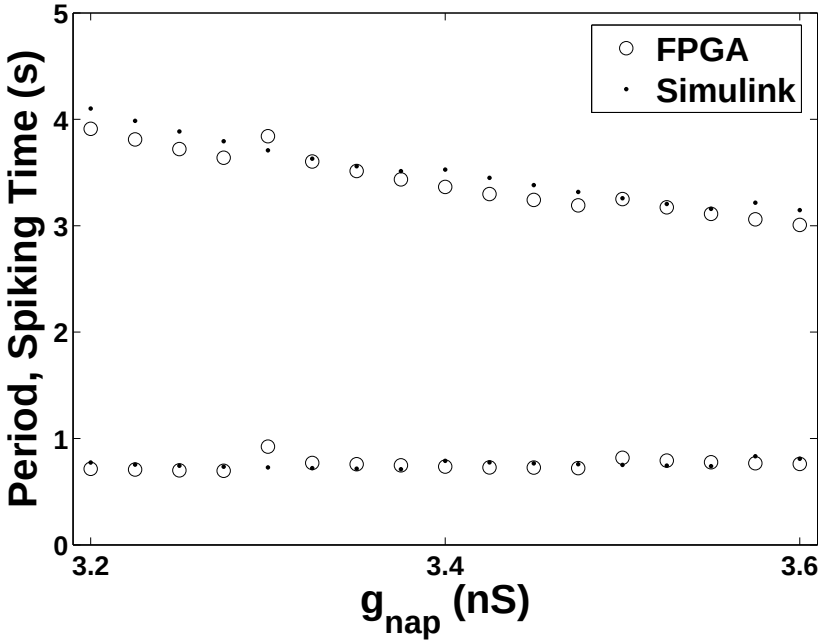


C.

Figure 28: Comparison between the MATLAB-SIMULINK model neuron and the FPGA model neuron—single-neuron bursting characteristics for sweeps of A. $\bar{g}_{\mathrm{NaP}}$, B. $\bar{g}_{\mathrm{Leak}}$, and C. E_{Leak}. The canonical values are shown by the dashed lines. The periods are the upper data, and the spiking times are the lower data.

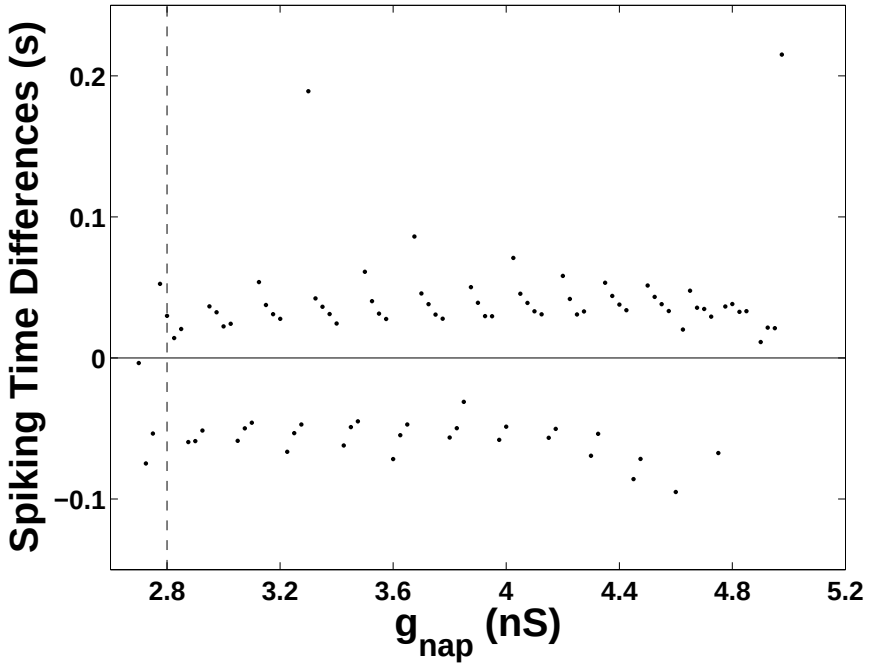


A.

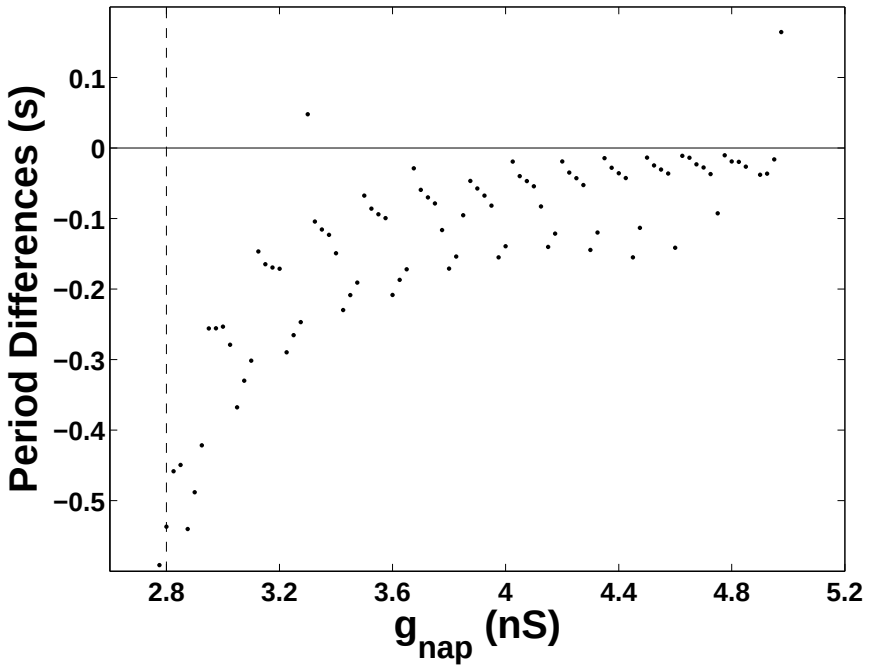


B.

Figure 29: Comparison between the MATLAB-SIMULINK model neuron and the FPGA model neuron—examination of the quantization error. The canonical value is shown by the dashed line. The periods are the upper data, and the spiking times are the lower data.



A.



B.

Figure 30: Comparison between the MATLAB-SIMULINK model neuron and the FPGA model neuron—examination of the quantization error. The canonical values are shown by the dashed lines. The data show the differences between the MATLAB-SIMULINK model neuron and the FPGA model neuron. These differences (quantization errors) are clearly systematic.

and the quantization error that was magnified at the $\bar{g}_{\mathrm{NaP}} = 3.3$ nS point in the previous figure can also be seen with this data. An alternative way to view (or to use) this data is to switch the axes to determine the value of an intrinsic parameter that is necessary to obtain some bursting characteristic; for example, if an average spike frequency of $20-25$ Hz is required, then E_{Leak} can be set in the range of $-58.5 - -59$ mV. As can be seen from all of these figures, the single-neuron bursting data from the FPGA model neuron matches closely to that of the MATLAB-SIMULINK model neuron.

2.3.6 Performance of the FPGA Model Neuron

Table 13 shows the most important FPGA-related factors that affect the performance of the simulations of the FPGA model neuron.[13] In our original design, we down-sampled the N V_{mem} outputs, combined them in time division demultiplexor blocks, and transferred that data from the FPGA and into MATLAB for analysis using gateway out blocks; this data transfer is the biggest bottleneck in the system. These settings required approximately 15 minutes of real time to complete a 30-second FPGA simulation, which was an unacceptable result if we were to obtain the amount of data that we desired to study a heterogeneous network. As a result, we needed a much improved method for taking the data off of the FPGA. Note that in free-running mode, the FPGA could generate a clock rate of 20 MHz, but that rate was not applicable for our setup.

The best way to improve the performance was to reduce the amount of data that was transferred from the FPGA to MATLAB. To do so, we implemented a peak detector algorithm that only transferred peak V_{mem} and time values to MATLAB using a FIFO block, which through its write enable pin is able to read specific values and ignore all of the rest. With these two major changes to the output implementation (the algorithm and the hardware), the data transfer rate was significantly reduced from about 10^4 samples per second to less than five percent of that sampling rate (Figure 32). The issues with this approach were that a complete view of V_{mem} was not possible and the design would not compile for 40 neurons (because of the size of the design coupled with the interconnect requirements),

[13] Clearly, the available resources and processing power of the PC also combine to play a critical role.

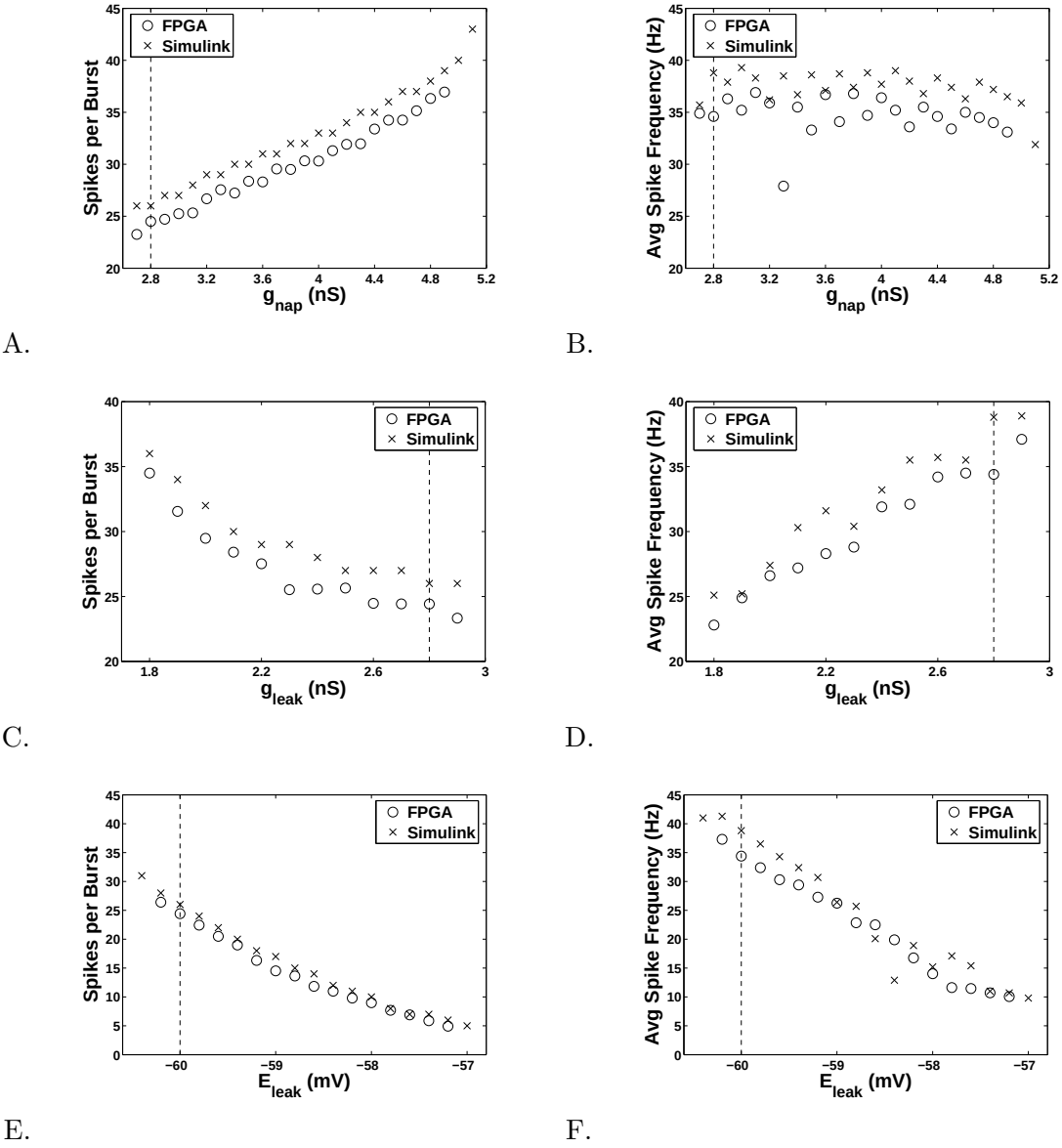


Figure 31: Comparison between the MATLAB-SIMULINK model neuron and the FPGA model neuron—single-neuron bursting characteristics for sweeps of A. and B. $\bar{g}_{\mathrm{NaP}}$, C. and D. $\bar{g}_{\mathrm{Leak}}$, and E. and F. E_{Leak}. The canonical values are shown by the dashed lines.

Table 13: FPGA model neuron—performance factors and settings.

neurons, FIFO blocks (N)	36
fixed step size (t_s)	9.9 μs
sampling rate	1 / (10 $\times$ step size) $\approx$ 10 kHz
V_{mem} output precision	16 total bits (8 fractional bits)
parameters to set	variable

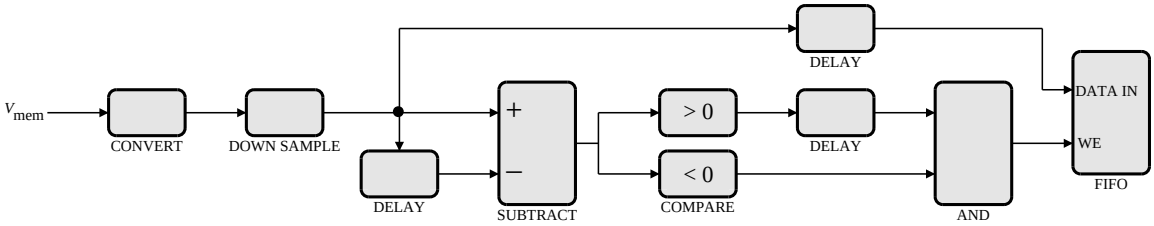


Figure 32: FPGA model neuron—peak detector circuit. The lightly shaded blocks represent the mathematical operations (convert, down sample, delay, subtraction, comparison, and). The convert block reduces the precision of V_{mem} from 51 bits to 16 bits. The down sample block reduces the sampling rate from about 10^5 samples per second to about 10^4 samples per second. The output of the and block controls the write enable pin of the FIFO block, which has a depth of 512.

which required us to reduce the network size to 36 neurons; these two issues were acceptable tradeoffs for the enormous performance gains that resulted.

With the new design, the largest performance variable was the number of parameters that required setting. A 100-second FPGA simulation required approximately 40 seconds to complete when a minimal number of parameters were set (< 100) and required approximately 70 seconds to complete when all of the parameters were set (≈ 1500). Note that these times were independent of the type of output (*e.g.*, bursting, tonic firing, silence). Using this later time, we measured the performance of the FPGA model neuron in terms of a real-time factor according to

$$\text{real-time factor} = N \times (\frac{\text{simulation time}}{\text{real time}}). \tag{43}$$

For our design, the real-time factor is about 50, which is an acceptable result because it satisfies our goal of having 40 neurons running with a real-time factor of 40.

A template of the five primary steps of the MATLAB code that we used to run the simulations is shown below:

```
%Step 1. Initialize the variables and model parameters:
simulationStopTime = 100;
modelParameter1 = 1;
modelParameter2 = 1;
...

%Step 2. Initialize and start the simulation using a Simulink command:
set_param(gcs,'StopTime',simulationStopTime)
set_param(gcs,'SimulationCommand','start')

%Step 3. Set the model neuron parameters using Randy Weinstein's command:
setparamhw('BurstNet','modelParameter1',modelParameter1)
setparamhw('BurstNet','modelParameter2',modelParameter2)
...

%Step 4. Execute a while loop while the simulation is running:
while get_param(gcs,'SimulationStatus') == 'running'
   pause(1)
end

%Step 5. Load, reconstruct, analyze, and save the data:
load Vmem1.mat
load Vmem2.mat
...
```

Step 3 is the critical step that affects the performance—the more parameters that are set, the longer the simulation as noted in the previous paragraph. Step 4 is required because two threads exist—the MATLAB m-script and the FPGA simulation. Without the pause, the MATLAB m-script will consume all of the resources, but with the pause, resources will be shared between the two threads. In Step 5, the reconstruction of the data is required because only the peak values and times are transferred between the FIFO blocks and into MATLAB for analysis. The analysis of the data will be explained in Chapter 4.

The following array is required to set the N^2 synaptic parameters in Step 3:

$$\bar{g}_{\text{syn}} = \begin{bmatrix} 0 & \bar{g}_{\text{syn}}(1,2) & \bar{g}_{\text{syn}}(1,3) & \dots & \bar{g}_{\text{syn}}(1,16) \\ \bar{g}_{\text{syn}}(2,1) & 0 & \bar{g}_{\text{syn}}(2,3) & \dots & \bar{g}_{\text{syn}}(2,16) \\ \dots & \dots & \dots & \dots & \dots \\ \bar{g}_{\text{syn}}(36,1) & \bar{g}_{\text{syn}}(36,2) & \bar{g}_{\text{syn}}(36,3) & \dots & 0 \end{bmatrix} \tag{44}$$

where $\bar{g}_{i,j}$ is the synaptic weight from Neuron i to Neuron j. Note the following two constraints involving symmetry: (1) $\bar{g}_{i,i} = 0$ will inhibit the neurons from synapsing on to themselves. (2) $\bar{g}_{i,j} = \bar{g}_{j,i}$ will result in the symmetrical matrix that is required for the homogeneous configuration. For the $N = 36$ implementation, Neurons $1 - 18$ formed one half of the HCO, and Neurons $19 - 36$ formed the other half. The construction of this synaptic-weight matrix, derived from Eq. (9) and Eq. (10), results in the following: (1) $\bar{g}_{1,j}$ (row) are the synaptic weights from Neuron 1 to all of the other neurons, and these synaptic weights are multiplied by $s(1)$, which is a function of $V_{\text{mem}}(1)$. (2) $\bar{g}_{i,1}$ (column) are the synaptic weights from all of the other neurons to Neuron 1, and these synaptic weights are aligned in time for a single computational step in the FPGA. As a programming note, each of the $N = 36$ rows of the synaptic-weight matrix are specified separately in MATLAB.

2.3.7 FPGA Implementation Design Decisions

Although an FPGA implementation is less complex, more stable, more accurate, and more flexible than a silicon-based analog one, we were still confronted with a number of issues with the FPGA platform. Because floating-point registers use more space and require more operations (*i.e.*, increase in latency) and because we were resource constrained, we only used fixed-point registers. As a result, we could not change orders of magnitude, and we encountered quantization error, although some amount of quantization error will always be present in our nonlinear system regardless of the register type. Using fixed-point registers, we were required to determine range and resolution specifications for all of the model parameters and signals throughout the system and were subsequently faced with a standard tradeoff—parameter precision/accuracy/resolution versus register type/usage/size. In addition, by efficiently determining the required number of bits, we were careful to not inadvertently

introduce excessive quantization error. For example, because the τ_x term in Eq. (2) cannot be zero, its register must contain a nonzero value. For $|(V_{\text{mem}} - \theta_x)| >> 2\sigma_x$ (from Eq. (4)), τ_x gets very small, but we simply limited the value of τ_x to the most efficient precision (18 bits). Last, we did not use a variety of ODE solvers; instead, we implemented a brute-force method by performing fast operations using the simple Euler algorithm. The next section describes the methodology and design flow employed by Weinstein.

2.3.8 Methodology and Design Flow

The methodology we used included an off-the-shelf FPGA development board interfaced through the Xilinx (San Jose, CA) System Generator, a graphical front-end development environment that is a component of MATLAB and SIMULINK.[14] Specifically, the design flow utilized System Generator v8.1 within MATLAB v7.1. System Generator is a SIMULINK blockset that provides a set of library blocks directly translatable into hardware constructs. These library blocks include math operators (adders, subtractors, multipliers), logic operators (multiplexors), and various forms of memory (single-bit and shift registers and logic-based and block RAM). System Generator translates the model into VHDL code and executes the associated Xilinx ISE v8.1i tools. The VHDL code is synthesized into the primitives on the FPGA, then placed, routed, and finally converted into a bitstream. This bitstream is programmed on the FPGA, providing its unique configuration for that particular model. System Generator then constructs a harness for simulating the model within SIMULINK. This entire process takes tens of minutes to hours, depending on model complexity.

The Xilinx XtremeDSP series of Virtex-II and Virtex-4 development boards were used.[15] The XtremeDSP development board is a repackaged Nallatech (Glasgow, UK) BenONE PCI carrier board containing a BenADDA module. The module contains the user programmable FPGA, either a Virtex-II XC2V3000 or a Virtex-4 XC4VSX35, dual 105 MSPS analog-to-digital converters, and dual 160 MSPS digital-to-analog converters. Note that these

[14] The material in this section was taken from a paper by Weinstein, Reid, and Lee that was accepted by IEEE Transactions on Neural Systems and Rehabilitation Engineering [122].

[15] Our work was customized to the Xilinx models of FPGA boards. Alterra (San Jose, CA) is the primary alternative.

particular development boards and design tools undergo version changes over time requiring minor revisions to the design flow.

Co-Simulation. The simultaneous execution of a model using both SIMULINK blocks and directly on the FPGA is termed co-simulation. The co-simulation environment is provided as a means for interacting with the model while it is executing. There are two different clocking modes—a full-speed, free-running clock that can be set to a number of standard clock frequencies and a slower, simulator-controlled, single-cycle clock. In general, the free-running clock will provide enhanced performance; extracting data at the maximal data rate via software, however, is difficult. In contrast, having SIMULINK control the clock provides full interaction and observability albeit at a loss of performance.

Several blocks are available for use with co-simulation. The gateway in and gateway out blocks provide one input or output, respectively, to the system. In single-cycle mode, the clocks are synchronized such that inputs are immediately available to the system at the next clock cycle and outputs are immediately available without delay. In free-running mode, data is transferred by best effort often with significant loss of data.

With the release of System Generator v7.1, shared memory blocks were introduced as an improved means of transferring large quantities of data. With shared memory blocks, a link is generated at run-time between a block of RAM on the FPGA and an associated memory buffer implemented in software. This allows the continuous monitoring and overwriting of values within the memory buffer. Two modes are available—a locked mode allows block transfers to and from the buffers via DMA and a standard mode makes no guarantee with respect to contention. We used the shared memory blocks in the standard mode for low-speed parameter updates while the FPGA was free-running.

Parameter Database. The assisted approach presented here required a new abstraction to describe the components and quantities within the model. In general, neural models fit a basic framework consisting of a system of first-order nonlinear differential equations. Each of these equations comprises a state variable (*i.e.*, memory-based component) and the data-path (*i.e.*, computational component); the data-path is a function of other states, parameters, and global constants within the system. In certain cases, a single differential

equation is divided into intermediate calculations whereby a transient quantity is calculated for use in the same time step in a successive state or intermediate calculation.

We generated a simple database within the MATLAB environment to track these identifiers (*e.g.*, names of variables in the system) and quantities of interest (*e.g.*, parameter values, state initial values). Four main categories are represented: states, intermediates, parameters, and constants. A variety of information is appended to each entry in the database. This central repository of pertinent model information enables the simplification of the System Generator model. Because parameter values and initial values of states are stored in the database, a particular System Generator model does not have to store these values locally. In effect, the "construction" details are clearly separated from the "model" details. Further gains are made via auto-generation, which is described in Section 2.3.9.

In addition to parameter values and initial values of states, other model and entry-specific quantities are stored. For example, type information is stored in the form of fixed-point notation, specifying the sign, the number of total bits, and the number of fractional bits. Parameters, in particular, have several flags associated with them:

- *adjustable*: the quantity is adjustable via the parameter subsystem
- *visible*: the parameter is adjustable by the user rather than an internal control signal
- *dependent*: the parameter is adjustable via one or more "visible" parameters

These flags allow the definition of multiple classes of parameters of the system, some of which are left hidden from the user. Many parameters, such as a maximal sodium conductance, $\bar{g}_{\mathrm{Na}}$, can be *adjustable* and *visible*, implying that it is a quantity used directly as an operand in the FPGA and is tunable by the user. In contrast, the output subsystem uses the parameter subsystem to provide control signals to dictate which variable in the system is routed to an analog output (a feature of the XtremeDSP Development Board series). These control signals are not modeler-tunable parameters, but can be modified using a special software routine. In this case, the output select parameter will be described in the database as *adjustable* but not *visible*.

The derived parameter is another special case. For example, membrane capacitance, C_{mem}, should be tunable by the user, but because it appears in the denominator of the membrane-voltage equation and is not in a form that can be computed efficiently on an FPGA (would require a reciprocal), that parameter should not be adjustable in the parameter subsystem. Instead, the parameter $1/C_{\text{mem}}$ is a *dependent*, *adjustable*, but not *visible* parameter while C_{mem} is a *visible*, but neither an *adjustable* nor a *dependent* parameter. Full support, in the form of functions for all of these derived parameters, is further enabled by the database.

As shown in previous work [41] [121], models are often pipelined to increase the utilization of an implemented data-path. For example, in a particular model, a 20-stage pipeline for the voltage state corresponded to 20 compartments or 20 neural models. The voltage state in this case can be thought of as a vector quantity consisting of 20 initial values. This work also suggested the use of circular buffers to store parameter values. Our parameter database, however, enabled both states and parameters to take on vector quantities to ease the implementation. These circular buffers were preloaded with parameter values offset by addresses based on the total insertion delay in the pipeline. The database stored the appropriate offset value to aid in construction of these buffers.

2.3.9 Auto-Generation of the Infrastructure

By using the parameter database, a large portion of the infrastructure can be readily auto-generated, freeing the modeler to focus on the modeling task at hand.[16] A variety of tools were generated in MATLAB focusing on three particular components—the state subsystems, the parameter subsystem, and an output subsystem to interface the analog outputs. These subsystems were created as dynamically linked SIMULINK subsystems, such that when the database was altered, a simple command would automatically update the library blocks and the effected models.

State Generation. As described in our previous work [121], multiple ways are possible to design the states and the differential-equation solver depending on the requirements. When

[16]The material in this section was taken from a paper by Weinstein, Reid, and Lee that was accepted by IEEE Transactions on Neural Systems and Rehabilitation Engineering [122].

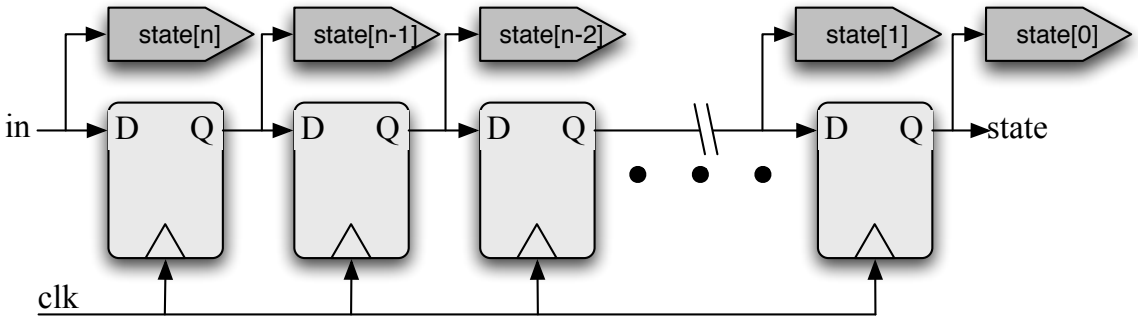


Figure 33: FPGA auto-generation of the infrastructure—state-storage subsystem. The lightly shaded blocks represent a register chain and are built using System Generator register blocks. The darkly shaded blocks represent SIMULINK from blocks that link with the corresponding SIMULINK goto blocks in the state read subsystem. For the described $N = 36$ neuron model, N registers and $N + 1$ from blocks are required.

multiple models are reusing the same data-path, we suggested to implement the states as a sequence of registers and to add "taps" to connect the registers to the inputs of other portions of the data-path. In an alternative case where the simulation contained only one model of interest, one register is used and clocked at the slower overall sample period. Both of these scenarios are auto-generated by the tools.

The PBC neural population [8] [9] used in our work consisted of 36 neurons. Therefore, a 36-stage pipeline was chosen, and all states were defined as vectors of length 36. Auto-generation of the states consisted of the creation of two library blocks per state. One library block is used for the storage of the stages as shown in Figure 33. The state-storage subsystem can optionally include the Euler-integration circuitry or those options can be relegated to the user-defined data-path. The other library block is a compile-time configurable tap into the shift register (see Figure 34).

The state-storage system was simply a shift register of length equal to the number of simultaneous models (36 in this case). A shift register was made from a single 36-cycle latency delay block. While area efficient, this approach had two drawbacks. First, System Generator did not allow initial values to be stored in the delay block, causing additional initialization logic to be required. Second, internal states were not accessible within a shift register; data was instead only available at the output of the chain. The solution was to form a shift register based on individual register blocks. These registers could be initialized and

allowed each output to be accessible. The drawback of this approach was the significant area resources required for each register. For example, a 36-stage pipeline with a 20-bit wide state variable consumed approximately 30 slices using delay blocks and 400 slices using register blocks.

The auto-generated state-read subsystems were designed to tap a particular position in the shift register. Each state-read block was parameterized by an offset value which was chosen to be equivalent to the total delay (latency in clock cycles) from the input, or leaf of the tree, to the output, or root of the tree (see Figure 25 where the h and the V_{mem} state-read blocks have an offset of 3 and 15 cycles, respectively, corresponding to the total delay from the state-read block to the state-storage block). This value is equal to the sum of the latencies of each operation that a particular input must propagate through to reach the output. By carefully designating these delays for each input, multiple models can be simulated in lock step without interference. Note that at each clock cycle, every model that simultaneously uses the data-path is accounted for within the system. Each instance of the model is delayed from the root of the tree, which subsequently decreases per cycle, eventually returning to the end of the queue for the next cycle.

The read subsystem utilized a multiplexor (see Figure 34), a primitive that chooses the appropriate output according to the select input. This was determined at compile-time (*i.e.*, before synthesis). Because the synthesis tool recognized that the select line of a multiplexor is a constant, the block was reduced to a single bus connecting input with output. Therefore, no resources were used in the instantiation of this block.

System Generator placed a hard limit of 32 on the number of inputs to a multiplexor. To accommodate pipelines exceeding 32 stages (for models with over 32 units utilizing the same data-path and up to $32^2 = 1024$ units), a two-level multiplexor scheme was employed. A total of $N_{\text{mux}} = \lceil d/32 \rceil$ first-level multiplexors were generated and fed into an N_{mux}-input multiplexor. The select line of each multiplexor became a function of the compile-time parameter, dictating the appropriate latency at the leaf of the tree.

This auto-generated state subsystem was a valuable contribution to the modeling process as it eased a common class of construction-related model modifications. Often, more or less

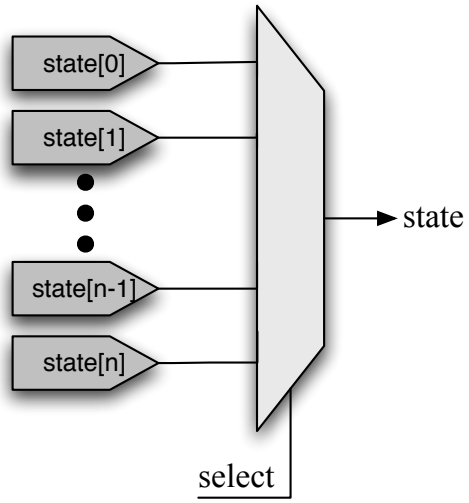


Figure 34: FPGA auto-generation of the infrastructure—state-read subsystem. This library subsystem block is unique per state and is instantiated for each read of the state throughout the model. The lightly shaded block represents a System Generator multiplexor block, which selects the appropriate state. The select line of the multiplexor is set at run-time and corresponds to the insertion delay required at the read block. Each darkly shaded block corresponds to a SIMULINK from block, which is linked to a goto block in the state-storage subsystem.

models are desired in a simulation (*i.e.*, the neuron pool is enlarged as the population size increases, the dendritic tree is enlarged or further subdivided, or a neural circuit model grows in complexity). Assuming the model does not shrink such that the total number of models is less than the maximum latency through the data-path, the number of models can be adjusted via a change in the parameter database and a regeneration of the state subsystems, with no change required to the actual user developed data-path. Additional changes to the parameters will also likely be necessary, but will similarly require little or no user modification to the model.

Parameter Generation. The parameter generation tools provided the full infrastructure to handle on-the-fly parameter tuning within the model. Based on a memory interface, they also allowed for optional control registers to set internal states of the model and, for example, allow the model to start, stop, and reset. The parameter subsystem (see Figure 35) utilized a System Generator shared memory block to store every quantity requiring dynamic modification.

A memory map was first created that consisted of each scalar and vector quantity in the

system. For our model, the memory map contained 1694 parameters (47 vectors of length 36 plus 2 scalars) for this subsystem. The majority of those parameters, 1296 in total, represented the synaptic weights. Four conductance vectors representing the maximal ionic conductance of a fast inactivating sodium current, a persistent sodium current, a potassium current, and a leak current for all 36 neurons accounted for 144 parameters. Individual leakage reversal potentials and excitatory/inhibitory synaptic reversal potentials per pre-synaptic neuron accounted for 108 parameters. In addition, two internal parameters stored in the memory map were used in the analog output subsystem.

The data-path did not directly read from this large memory, but instead read from local registers that are constantly updated by this RAM. A counter asynchronous to the data-path continuously cycled through every address in the memory. Simultaneously, a single token was propagated through a circular shift register of length equal to the depth of the RAM. This token became an enable for one of two storage elements. In the case of a scalar value, this token fed the enable of a register that was updated with the value from the RAM. For a vector, more circuitry was employed to refresh a dual-port RAM with depth equal to the length of the vector (32 in this example). This token triggered a counter (from 0 to 31) and the write-enable signal of one port of the dual-port RAM. The token stayed active for 32 cycles, sufficient for all values to be updated within the vector RAM.

Because all parameters were not of the same size or type, logic was added for System Generator to ensure that the correct types were used in synthesis. The output type of the shared memory block in Figure 35) was set as an unsigned integer with width equal to the widest parameter value. Each integer representation, Z, was initialized by

$$Z = \begin{cases} \text{round}\left(x \cdot 2^f\right) & x \geq 0 \\ \left(!\ \left(\text{round}\left(-x \cdot 2^f\right)\right) + 1\right)\ \&\ \left(2^n - 1\right) & x < 0 \end{cases} \tag{45}$$

where x is the real-valued number, f is the number of fractional bits, n is the total number of bits, and the round function rounds the real number to the closest integer. Negative numbers must be first negated, then converted to an integer, whereby the 2's complement (inversion of all bits marked by ! unary-operator plus 1) is masked (AND binary-operator

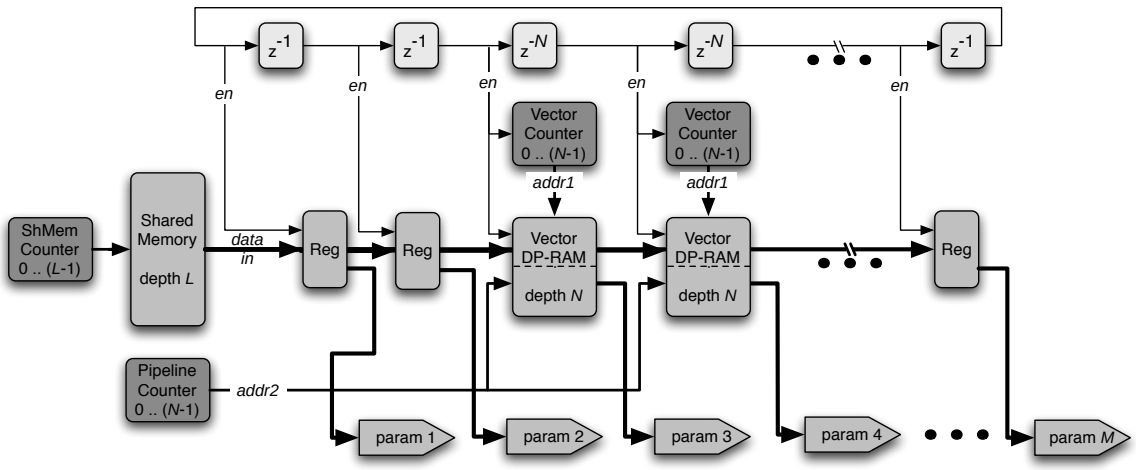


Figure 35: FPGA auto-generation of the infrastructure—parameter subsystem. This auto-generated subsystem enables the on-the-fly tuning of scalar and vector parameters via a Shared Memory block. A total of L values representing a combination of M scalar and vector parameters are supplied to a N-stage pipeline. The lightly shaded blocks at the top represent the token ring structure consisting of a bit-wide register/delay chain. Delays are either a single cycle for a scalar quantity or an N-cycle delay for a vector quantity. The darkly shaded blocks represent counters in the system. Thin traces represent control signals such as enables to the vector counters and a write enable to the first port of the dual-port vector RAM block. Medium traces indicate address busses to each of the RAM blocks. The thick traces depict those busses containing parameter data.

indicated by &) by a sequence of n-digit binary sequence of 1's. The output must go through a conversion phase before reaching the input port of the register or RAM. This consists of a System Generator convert block changing the type to an unsigned integer at the appropriate number of bits for the parameter, cascaded with a System Generator reinterpret block to set the signedness and the fractional point of the parameter. The convert block can change a value while the reinterpret block can only alter the representation of the signal. In this case, the convert block acts to truncate the unused most significant bits. The auto-generation tools only placed these blocks as necessary for the parameter.

The outputs of this parameter subsystem fed the data-path. For scalar values, a register enabled the data to always be available. For vectors, the situation was more complex. The other port of each dual-port vector RAM was addressed by a counter equal to the vector, or pipeline, length. This counter was synchronized with the sample period of the entire data-path. Because parameters needed to be delayed to correspond to their relative insertion points within the data-path, the vectors were initialized and continuously refreshed in a sequence that was circularly rotated by the number of cycles equal to the desired offset (see [121] for more on circular buffers supplying parameter values to a pipelined data-path).

The parameter RAM suffered high fanout as it supplied the input of every vector RAM and scalar register in the parameter subsystem. While neither implemented nor required in the past, the high fanout can readily be remedied through the use of a register tree to balance the signal. The initial value of the token can then be changed to align with the delays added to the data signal.

Output Generation. The output subsystem was the third of the three auto-generated components (see Figure 36). This system linked the outputs of the data-path with the analog outputs available on the XtremeDSP development board. This subsystem generator overcame two main limitations within the included System Generator DAC blocks: 1) only two analog output channels were available although more signals were often wished to be viewed, and 2) analog outputs required a signed type of 14 total bits with 13 fractional bits.

Variables were designated and assigned to a particular analog output in the parameter database. The subsystem generator constructed a two-level multiplexing scheme to route

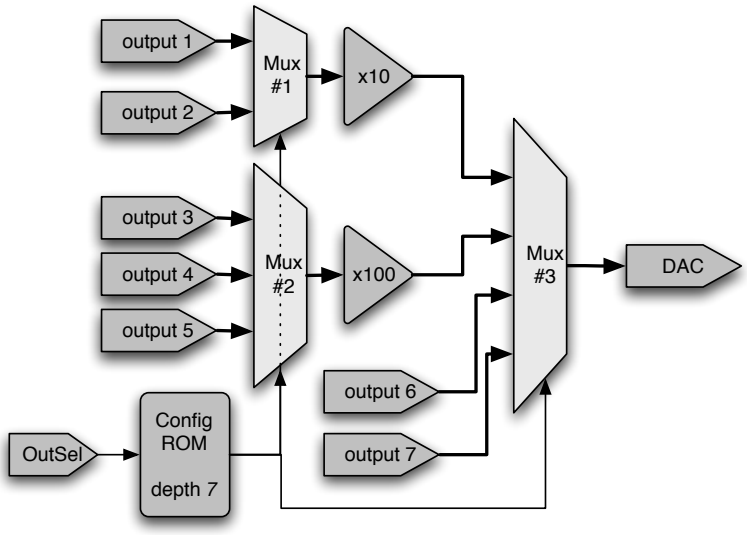


Figure 36: FPGA auto-generation of the infrastructure—output selector subsystem. A stereotypical structure is depicted with 7 variables. Outputs 1 and 2 are scaled by a factor of 10, outputs 3–5 are scaled by a factor of 100, and outputs 6 and 7 are unscaled. The OutSel signal, driven by the parameter subsystem, has a range of $0-6$ and sets the address line to the configuration ROM. In this example, the ROM outputs a 5-bit signal, of which one bit is dedicated to Mux #1, two bits are routed to Mux #2, and the remaining two bits connect to Mux #3. Mux #1 and #2 make up the first-level multiplexors, and Mux #3 is the sole second-level multiplexor.

the appropriate variable to the analog output. The multiplexors that comprise the first level were separated by type. Because the variable type might not align with the required type (signed, 14 total bits, 13 fractional bits), those variables with the same type were grouped and scaled collectively at the output of the multiplexor. Two scaling modes were supported: a zero-hardware power of 2 and a power of 10, which required the use of multiplier blocks. Each scaled or non-scaled output was then selected via a second-level multiplexor (see Figure 36).

Control logic was required to select the appropriate input per multiplexor. The outputs were enumerated and a control signal specifying the desired variable to be routed to the output was assigned as an input to the system. Two additional *adjustable*, but not *visible* parameters (*DACSelA* and *DACSelB*) were added to the parameter database so that the existing infrastructure could aid in run-time configuration of the output subsystem. The output select signal per analog output is used as an address into a configuration ROM

(read-only memory), which drove each of the select lines for all of the multiplexors in the subsystem. In particular, System Generator slice blocks were used to separate the appropriate subset of signals within the configuration ROM (see Figure 36) output, which was then routed to the multiplexors.

CHAPTER III

ANALYSIS OF BURSTING REGIONS

Rhythmic pattern-generating networks are essential to locomotion and other complex oscillatory neural behaviors, and the systems that produce and control these stereotyped movements have been optimized to be both energy efficient and dependable [79] [50]. As a result, these networks provide a platform for studying a critical set of biological control paradigms and inspire the research into engineered, hardware-based systems that exploit these underlying principles [81]. Because the neuron is the fundamental building block of these systems that are used to emulate rhythmic patterns of activity such as bursting,[1] the study of neural models is integral to this work, and the identification of a bursting region in a vast multi-parameter space is critical to our study of rhythmic pattern-generating networks.

Evaluating a representative number of neural outputs from a vast parameter space is a daunting task. In a seminal study, the relationships between neural behavior and parameter values were analyzed using HH models [55] in which repeated uses of a stochastic search algorithm were implemented [33]. Subsequent work that examined the roles of parameters in conductance-based neuron models includes one study that showed that parameter averages over multiple samples can fail to characterize a system [40] and another study that evaluated samples from a database of approximately 1.7 million single-compartment model neurons that were generated by varying maximal conductances [94]. Because we are considering more parameters and more values of each parameter than these previous studies, we require a different approach for exploring a parameter space of a conductance-based model. To get an idea of the magnitude of the parameter space, suppose that the data for one million different outputs had been collected for a 10-parameter space. This seemingly large set

[1]The term bursting lacks a formal definition, but in common usage it refers to a neural output that is characterized by periodic repetitive activity of action potentials punctuated by periodic episodes of silence. A bursting region is a contiguous region in parameter space in which bursting occurs. See Izhikevich for an exhaustive description [59].

of data represents a parameter-space sampling of only four different values per parameter ($4^{10} \approx 1$ million). Increasing the sampling to, say, 10 values per parameter results in 10^{10} combinations. If 10 seconds of data were taken at 100 times real time at each of these 10 billion points in parameter space, then the time required to accumulate the data would be approximately 32 years. Therefore, we clearly can only sample the parameter space, and we must derive our conclusions from this relatively small sampling.

A key macro-scale organizational principle of the human brain is that neurons, the fundamental units of biological nervous systems, have similar architectures but produce vastly different actions resulting in a remarkable range of human behavior [62]. This reusability of architectures to produce multiple actions is also evident in animal nervous systems such as those that underlie rhythmic motor-pattern generation. These neuronal networks allow the cellular and synaptic properties of individual neurons to be altered, thus achieving a variety of motor patterns [93]. Semiconductor-based physical systems exhibit real-time behavior, consume low power, and are small and portable. In addition, the physics underlying the movement of charge in these real-world aVLSI systems is similar to that of living neurons giving neurobiologists an effective medium in which to study complex neurophysiological problems [81]. Furthermore, science that exploits these silicon properties provides future promise in such fields as autonomous robots and neural prosthetics. We, however, desire a more immediate application of this technology such as using the silicon-neuron architecture [106] as a tool to test theories under realistic and real-time conditions—that is, whether living neurons with similar flexible architectures can be used in a variety of behaviors. In addition, we also want to explore the ideas of collective computation among many imprecisely matched elements (*i.e.*, with heterogeneity imposed on the system).

Motivated by this background information, we formalized tests using the computational model (Section 2.1) that involved search techniques in vast parameter spaces and determinations of contiguous regions of bursting space. With that knowledge, we used the silicon implementation (Section 2.2) to explore the relationships between disparate models and to understand the role of intrinsic heterogeneity, and that exploration was a first step in the study of the role of heterogeneity in rhythmic networks of neurons. This chapter describes

the single-neuron tests that we completed to accomplish those goals, and the framework of this chapter is as follows: In Section 3.1, we will describe our use of the MATLAB-SIMULINK model neuron to demonstrate a parameter-space search algorithm using the PBC neuron (Section 2.1) because, as explained previously, a brute-force method for searching its vast parameter space is unpromising. We will show that a simple cost function, whose inputs are derived from the frequency content of the neural output, and a stochastic gradient descent-type algorithm can be used to locate bursting regions and that these regions are contiguous. In Section 3.2, we will demonstrate the flexibility of the architecture of our silicon model neuron by implementing three disparate conductance-based neuron models using a single integrated circuit. More importantly, we will show that two of these dynamically different models represent points in a contiguous bursting space that spans between the two models. In Section 3.3, we will continue our work with the silicon model neuron to study the intrinsic heterogeneity in a population of isolated neurons that were fabricated on a single integrated circuit. To quantify this heterogeneity, we will demonstrate our techniques for mapping the input and output parameter spaces of each of the neurons.

3.1 A Parameter-Space Search Algorithm Using a Computational Model

We demonstrated a parameter-space search algorithm using the computational model of the single-compartment, conductance-based PBC neuron that was described in Section 2.1.[2] To classify bursting (the desired behavior), we used a simple cost function whose inputs are derived from the frequency content of a neural output. Our method involved the repeated use of a stochastic gradient descent-type algorithm to locate parameter values that allowed the neural model to produce bursting within a specified tolerance. We demonstrated good results, including those showing that the utility of our algorithm improved as the predefined allowable parameter ranges increased and that the initial approach to our method is computationally efficient. The framework of this section is as follows: In Section 3.1.1, we will describe the search algorithm, including the cost function and the roles of the

[2]The material in this section was taken from a paper by Reid, Brown, and DeWeerth that was submitted to Biological Cybernetics in September 2006 [97].

pre-defined parameter ranges and standard deviations. In Section 3.1.2, we will describe our methods for obtaining and processing the neural output data. In Section 3.1.3, we will show our experimental results, including an analysis of the role of the pre-defined parameter ranges and an illustration of a single trial. In Section 3.1.4, we will conclude this section with a discussion of improvements that could be made to our algorithm.

3.1.1 The Search Algorithm and Cost Function

Our objective was to search a vast parameter space of a neural model to find combinations of parameters that achieve a neural output that meets our specifications. To achieve our goal, we used a stochastic gradient descent-type algorithm that determined the parameter values for each simulation. This parameter-selection process also included a cost function[3] that used frequency-domain information from the neural output, V_{mem}.

Cost Function. We defined three frequency components as the inputs to the cost function, and its output gave an indication of the "frequency distance" between an experimental and target waveform—that is, one that results from the published parameter values. The goal of our search routine was to minimize the value of the cost function, which can be presented in two forms

$$C(x_f, y_f) = (y_f - x_f)^2 + (\frac{y_f}{x_f} - 1)^2 \tag{46}$$

$$= (y_f - x_f)^2(\frac{x_f^2 + 1}{x_f^2}) \tag{47}$$

where the subscript $f \in \{\text{low}, \text{mid}, \text{high}\}$, x_f is the actual frequency content of V_{mem} in the specified range, and y_f is the desired, or target, frequency content. The total cost function, C, is then given by the sum of the three subcomponents:

$$C = \sum_f C(x_f, y_f). \tag{48}$$

The functional, or conceptual, form of the cost function is given by Eq. (46). The first term is a squared error between the target and actual values and is commonly used in linear system analysis to find a minimum. Specific to our application, the second term is linearly

[3]Other names for this function are "energy function" and "loss function."

independent of the first and is used to determine the ratio between the target and actual values to force the cost function to "blow up" when the presence of a range of frequencies is negligible. The alternative representation given by Eq. (47) transforms the $(y_f - x_f)^2$ term into a new space determined by $(x_f^2 + 1)/x_f^2$. The significance of the different regions of parameter space is highlighted by this alternative form for two cases:

- For x_f of order one, the $(y_f - x_f)^2$ term dominates because the cost function is close to a minimum.
- For $x_f << 1$, the cost function effectively becomes $1/x_f^2$ and is inordinately far from a minimum, resulting in an extremely large cost function value.

Note that the cost function, $C(x_f, y_f)$, satisfies the following constraints:

- $C(x_f, y_f) \geq 0$.
- $C(x_f, y_f) = 0$ for $x_f = y_f$.
- $\frac{dC(x_f,y_f)}{dx_f} = 0$ for $x_f = y_f$.
- $C(x_f, y_f)$ is a continuous function.

These constraints mean that a larger cost function value implies a "farther distance" from the target output—as the cost function decreases, the output evolves smoothly into a representation of the target output. Note that $C(x_f, y_f)$ has no dependence on the history of the frequency content of V_{mem} or the solution path followed by the algorithm.

Setup. The process of determining the parameter values for each simulation is referred to as an evaluation, E, and the parameter values of the n parameters are given by:

$$\mathbf{P}_E = P_1, P_2, \ldots P_n, \tag{49}$$

and the value of the cost function is given by C_E. The algorithm continued until the value of the cost function fell below a threshold, $C_{\text{threshold}}$, or a maximum number of evaluations, E_{max}, was reached. The difference between the value of the cost function for evaluation $E - 1$ and the minimum value of the cost function for all prior evaluations is given by:

$$\beta_{E-1} = C_{E-1} - C_{\text{min}} \tag{50}$$

where $C_{\min} > C_{\text{threshold}}$.

To initialize the algorithm ($E = 1$), the starting value of a parameter was selected randomly from a uniform distribution based on its pre-defined range, and this range also provided hard limits on the parameter value for all evaluations. For $E > 1$, all parameters were moved simultaneously prior to each evaluation using the following difference equations:

$$\mathbf{P}_E = \mathbf{P}_{E-1} + (\mathbf{G}_E + \mathbf{M}_E) \tag{51}$$

where $\mathbf{G}_E$ is a set of zero-mean normally distributed values that uses pre-defined standard deviations for each parameter and $\mathbf{M}_E$ is a set of momentum terms given by two cases defined below. To simplify the following notation, we defined $\boldsymbol{\alpha}_E$ as the set of parameter differences relative to the previous evaluation ($\boldsymbol{\alpha}_E = \mathbf{G}_E + \mathbf{M}_E$).

Momentum Case I. If a new minimum cost function value was obtained ($\beta_{E-1} < 0$), then $\mathbf{M}_E$ was given by:

$$\mathbf{M}_E = m \cdot \boldsymbol{\alpha}_{E-1} \tag{52}$$

where the momentum factor, $0 < m < 1$, was a fixed gain that dampened the momentum expression (we used $m = 0.9$ for both cases). Because $\mathbf{P}_E$ is a function of $\boldsymbol{\alpha}_{E-1}$ and $m > 0$, the means of $\boldsymbol{\alpha}_E$ continued to move in the same direction relative to the previous changes in parameter values. For example, if a parameter value is given by λ and its value on the next evaluation is reduced by δ, then its value on the following evaluation will effectively be chosen from a normal distribution that has a mean *less than* $\lambda - \delta$.

Momentum Case II. If a new minimum cost function value was not obtained ($\beta_{E-1} > 0$), then $\mathbf{M}_E$ was given by:

$$\mathbf{M}_E = -m \cdot \boldsymbol{\alpha}_{E-1} \cdot f(\beta_{E-1}). \tag{53}$$

Because of the negative sign in Eq. (53), the means of $\boldsymbol{\alpha}_E$ moved in the opposite direction relative to the previous changes in parameter values. The momentum term for this case also included a sigmoidal function of β_{E-1} and is given by:

$$f(\beta_{E-1}) = \left(1 + e^{\frac{\mu_1 - \beta_{E-1}}{\mu_2}}\right)^{-1} \tag{54}$$

where the choices for the positive constants μ_1 and μ_2 resulted in the following approximate values:

$$f(\beta_{E-1}) \approx \begin{cases} 0.92 & \beta_{E-1} = 10 \cdot C_{\text{threshold}}, \\ 0.50 & \beta_{E-1} = 5 \cdot C_{\text{threshold}}, \\ 0.08 & \beta_{E-1} = 0.1 \cdot C_{\text{threshold}}. \end{cases} \tag{55}$$

This sigmoidal function modulated $\mathbf{M}_E$ and was critical when C_{E-1} was within an order of magnitude of $C_{\text{threshold}}$. Without the sigmoidal function, our method would not work because a cost-function valley, containing cost-function values less than the threshold, would be overshot.

Note that the use of the parameter differences, $\boldsymbol{\alpha}_{E-1}$, in the algorithm provided a history in the cost function and resulted in a form of momentum; that is, the current set of parameter differences was a function of the previous set of parameter differences. As a result, *all* sets of parameter differences included information from previous evaluations. A graphical representation of this algorithm is shown in Figure 37.

Parameter Ranges and Standard Deviations. The specified parameter ranges and standard deviations controlled the progress of the search. We employed a simple method to determine the range of a parameter—use a published value as the middle of a wide range that maintained biological relevance. We will show that the probability of success depended on these pre-defined ranges.

The standard deviations assigned to each of the parameters were effectively used as gains in the algorithm. The choices for these values were more arbitrary than those for the parameter range values, and the effects of these values were more complex. Some parameters might need a larger standard deviation to ensure adequate variability to move out of a valley in the cost function and into a different part of the solution space. This response, however, was likely to be detrimental if a solution is nearby. This tradeoff—parameter movement versus missed solutions—is standard in optimization problems and is described in the *no free lunch* theorems [132]. As an alternative to using constant standard deviations, an evolving standard deviation that decreased with $C_{\min}$ could be employed. This idea is referred to as *annealing* in minimization algorithms [113] as it is related to the

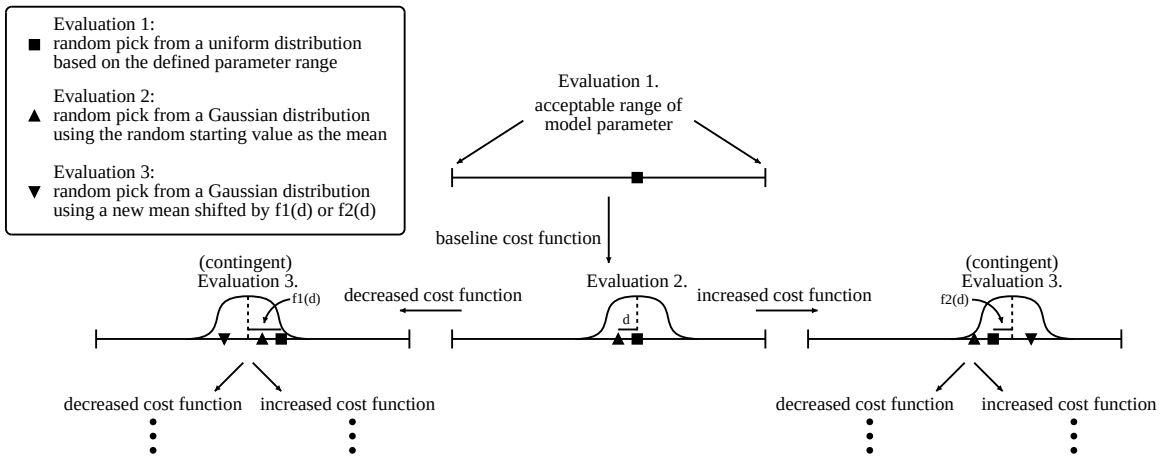


Figure 37: Parameter-space search algorithm—graphical representation of the algorithm used to determine a single parameter value for successive evaluations. For every evaluation, all parameters in the model are moved simultaneously using this method. Note the mean of the normal distribution and subsequent random pick in Evaluation 2. For the next evaluation, if the cost function decreases (*i.e.*, an improvement), then the mean of the normal distribution in Evaluation 3 (on the left) continues to move in the same direction as the previous move; otherwise, the opposite is true. $f1(d)$ and $f2(d)$ represent functions of the distance d (*i.e.*, the difference between the last two baseline values). The method used in Evaluation 3 repeats until the cost function falls below a threshold or a maximum number of evaluations has been reached.

process of slowly lowering the temperature of a hot pliable metal to change its shape. With our application, the larger the standard deviation (analogous to a high temperature), the more likely the current point in parameter space will move out of a local minimum to find a solution that meets our specifications (analogous to the desired shape).

3.1.2 Experimental Techniques

Most of the simulations were executed using MATLAB (MathWorks) Version 6.1 and SIMULINK Version 4.1 on a Dell Dimension 4300 personal computer with a 1.6 GHz Intel Pentium 4 processor and 1 GB of RAM running the Microsoft Windows 2000 operating system.[4] For a given set of model parameters, we ran a 60-second simulation of the model neuron using a variable-step solver in MATLAB. Because a fixed step size was required for the subsequent frequency analysis, we re-sampled the simulation data using a rate that was sufficiently

[4]Some of the later simulations were executed on a Sun Microsystems SunBlade 1000 running the Solaris Unix operating system. The processing power between this machine and the PC was similar.

large to accurately recover the neural spiking (*i.e.*, at least an order of magnitude greater than the maximum spike frequency).

To calculate a good estimate of the spectrum of a continuous-time signal, a periodogram method was used [88]. This method, which used windows to select a finite-length segment of samples from the data, involved a tradeoff—precision/accuracy/resolution versus computation time. The more windows used to estimate the frequency components, the better the resolution, but the more computation time that was required. Our algorithm required the frequency components to smoothly change between similar parameter sets; to achieve this and obtain a good tradeoff between precision and computation time, the number of samples should be twice as large as the number of windows, assuming a reasonable number of windows.

We used the `spectrum` command in MATLAB to perform the frequency analysis, and the frequency outputs from the command were used to plot a frequency-energy curve.[5] We merged the values across the frequency spectrum into three regions by simply summing the energy content in each of these regions, effectively reducing the resolution of the frequency-energy plot to three points. We chose the mid-frequency range based on realistic spiking frequencies, and the low- and high-frequency ranges were defined as the adjacent regions, resulting in the following frequency bins: (1) low frequency (< 10 Hz), (2) mid frequency (10–100 Hz), and (3) high frequency (> 100 Hz). Figure 38 is a cartoon of representative results from this FFT processing for a bursting output (in which frequency components exhibit continuous-like ranges) and for a tonic firing output (in which only the harmonics of the fundamental frequency have nonzero energy).

3.1.3 Results

In this section, we will show the results of the testing of our algorithm. We will first describe our process for locating hundreds of bursting points using our algorithm. We will then consider the roles of the pre-defined parameter ranges. We will highlight a trial and show how specific parameters progressed to a successful bursting output. Finally, we will

[5] Energy and power are general terms for the output of an FFT. The root-mean-squared voltage, V_{rms}, is the actual output for our case.

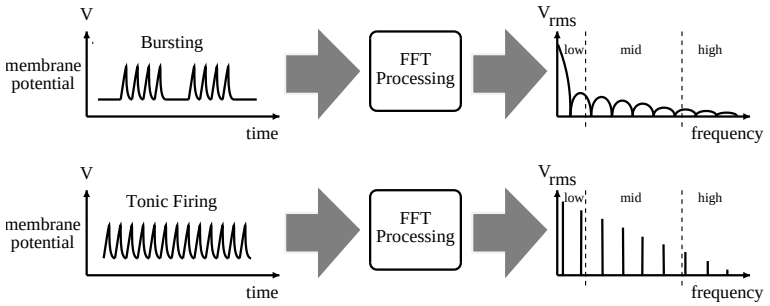


Figure 38: Parameter-space search algorithm—processing the data using the Fast Fourier Transform (FFT) to represent a time-based signal by its frequency components.

Table 14: Parameter-space search algorithm—pre-defined allowable ranges (standard deviations) for each of the 12 spiking parameters.

Reversal Potentials (mV)	
E_{Na}	50 ± 15 (2)
E_{K}	-85 ± 15 (2)

Maximal Conductances (nS)	
$\bar{g}_{\mathrm{Na}}$	28 ± 14 (1)
$\bar{g}_{\mathrm{K}}$	11.2 ± 5.6 (0.5)

Time Constants (ms)	
$\tau_n(I_{\mathrm{Na}})$	10 ± 5 (2)
$\tau_n(I_{\mathrm{K}})$	10 ± 5 (2)

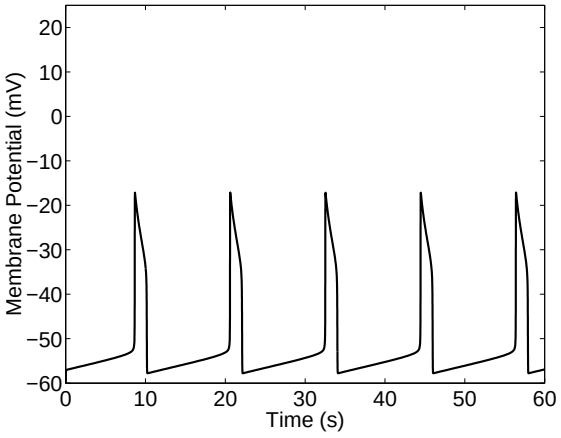
Half Maximal Voltages (mV)	
θ_q	-34 ± 15 (2)
$\theta_n(I_{\mathrm{Na}})$	-29 ± 15 (2)
$\theta_n(I_{\mathrm{K}})$	-29 ± 15 (2)

Slope Voltages (mV)	
σ_q	-5 ± 2 (0.5)
$\sigma_n(I_{\mathrm{Na}})$	4 ± 2 (0.5)
$\sigma_n(I_{\mathrm{K}})$	-4 ± 2 (0.5)

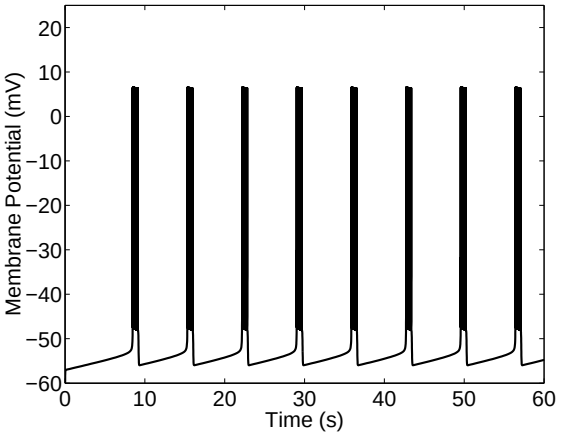
show that one contiguous bursting region contains all of the bursting points.

Setup. To simplify the search, we fixed the nine subthreshold parameters used in the Leak and NaP currents to their published values and only varied the 12 spiking parameters used in the Na and K^+ currents. As a result, the output of every bursting point that we found had the same underlying subthreshold oscillation that could be obtained by simply "turning off" the spiking currents—that is, by setting their maximal conductances to zero (*i.e.*, $\bar{g}_{\mathrm{Na}} = \bar{g}_{\mathrm{K}} = 0$ nS). The ranges and standard deviations of the 12 spiking parameters are shown in Table 14, and the published values are the midpoints of each of these ranges.

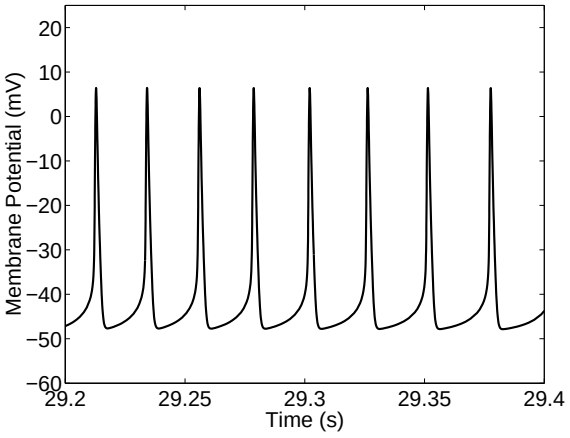
Using the published parameters shown in Table 2, we first recorded subthreshold oscillations and bursting activity (Figure 39). The target low-, mid-, and high-frequency outputs were 29.2 $\mathrm{V_{rms}}$, 12.4 $\mathrm{V_{rms}}$, and 10.7 $\mathrm{V_{rms}}$, respectively (for mathematical convenience, we multiplied the FFT outputs by 100 to use a larger $C_{\mathrm{threshold}}$). The target bursting also had a period of seven seconds, duty cycle of 10 percent, spiking frequencies of $20 - 60$ Hz, and



A.



B.



C.

Figure 39: Parameter-space search algorithm—data from the published parameters (see Table 2). A. Subthreshold oscillations ($\bar{g}_{\mathrm{Na}} = \bar{g}_{\mathrm{K}} = 0$ nS). B. Bursting. C. Spiking during the burst (approximately 40 Hz during this part of the burst).

26 spikes per burst.

We defined a *successful* trial as one that obtained a cost function value below 100 (*i.e.*, $C_{\text{threshold}} = 100$) with a value of 0 representing the target case. Note that this was an application-specific cost function; for example, sinusoidal inputs at the appropriate frequency and magnitude would have resulted in a minimum cost function value, but would not represent biological bursting. We specified the threshold value after considering the tradeoffs—obtaining a cost function value less than the cutoff that did not result in bursting (a false positive result) and obtaining a cost function value greater than the cutoff that did result in bursting (a false negative result). For our algorithm, no threshold value can perfectly classify all possible outputs. Consequently, a limitation of our method was that a biologically realistic bursting output that is not adequately characterized by the target time-based bursting attributes would not sufficiently comprise the specified frequency-domain information and would thus not be classified as a successful bursting output.

Initial Search. We ran 400 trials using the ranges shown in Table 14, with a limit of 500 evaluations for each trial ($E_{\text{max}} = 500$), and we found 357 bursting points (89 percent of the trials), which required 51 hours to complete. A plot showing the percent of successful trials within a specified number of evaluations is shown in Figure 40 (the upper bold data).

The data shown in Figure 40 indicate that half of the trials located a bursting point within 32 evaluations, and the inset of the curve shows that the percentage of successful trials saturates at about 90 percent, well before the 500 evaluation limit. More importantly, about three percent of the trials were successful on the first evaluation at the point determined by the initial random parameter selections. This percentage gave an indication of the size of the multi-parameter bursting hyperspace enclosed by the parameter ranges. In fact, given the relative size of this bursting hyperspace, a purely *random* search in this parameter space would yield better results for *any* number of evaluations (shown as the thin line in Figure 40). For example, in a random search case in which three percent of the evaluations are successful, half of the trials would locate a bursting point within 21 evaluations. Therefore, because our algorithm actually performed *worse* than random selections given the pre-defined parameter ranges, we had to consider the effect of changing

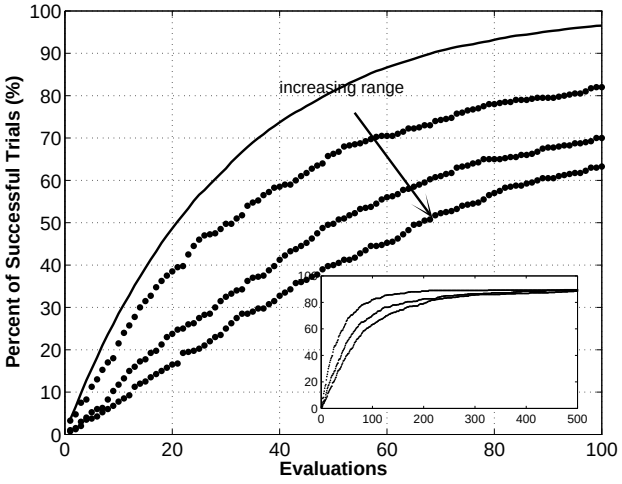


Figure 40: Parameter-space search algorithm—cumulative distribution of the percentage of trials that were successful within a given number of evaluations for various parameter ranges. The ranges, starting from the top, are 100, 120, and 140 percent of the parameter ranges shown in Table 14. The inset expands the results up to the maximum number of allowable evaluations. The thin line shows the results for random selections for the case in which the bursting region comprised three percent of the parameter space.

these ranges.

Evaluating the Effects of Different Parameter Ranges. We analyzed how the success of a trial was affected by the size of the parameter ranges in which the minimum and maximum values of each of the parameter ranges were changed by equal amounts. The results for these trials, shown in Figure 40, are for parameter ranges that are 100, 120, and 140 percent of the parameter ranges shown in Table 14. Four hundred trials were completed for each of these cases.

As expected, the success of a trial for less than 100 evaluations was dependent on the size of the pre-defined parameter ranges—a larger allowable hyperspace reduced the probability of success for a given number of evaluations during the first 100 evaluations of a trial. As shown in the inset of the figure, the probability of success after 500 trials is about 90 percent, regardless of the size of the allowable parameter ranges. Given by the success rate after one evaluation, the relative sizes of the bursting spaces for each of these cases are 3, 1, and 0.75 percent, respectively.

In a random search case in which one percent of the evaluations were successful (120% parameter ranges), our algorithm had a greater rate of success up to 154 evaluations (79% successful trials), and in a random search case in which 0.75 percent of the evaluations were successful (140% parameter ranges), our algorithm had a greater rate of success up to 234 evaluations (83% successful trials). These results indicated that the size of the parameter ranges played a critical role in determining when our algorithm was valid. For example, in the extreme case in which the pre-defined parameter ranges were virtually zero, our algorithm would not facilitate a search for bursting points. At some instance as the parameter ranges increase, however, our search algorithm becomes valid in a comparison to a random search. For our data, this occurred between the 100% and 120% parameter ranges. Consequently, given a small bursting region relative to the size of the multi-parameter hyperspace, we were encouraged by the results of our simple search algorithm. As the parameter ranges increase[6] and as the search algorithm improves (Section 3.1.4), our algorithm becomes more compelling (*i.e.*, the validity of our algorithm is related to the relative size of the bursting hyperspace). In the next section, we will quantify the size of the bursting region.

Quantifying the Bursting Region. To quantify the extent of the 357 bursting points found in the initial experiment that used the parameter ranges shown in Table 14, we defined a Euclidean distance measure, D_{E}, that can be used to determine the distance between any two points in this space.[7] To avoid adding the disparate units of conductances, time constants, and voltages, we normalized each of the parameters by defining the range of each parameter from 0 (minimum) to 100 (maximum). Therefore, the canonical value of each parameter had a normalized value of 50. The distance between any two points, $\mathbf{P}(a)$ and $\mathbf{P}(b)$, is given by the following:

$$D_{\mathrm{E}} = \sqrt{\sum_{n=1}^{12} \left(P(a)_n - P(b)_n\right)^2} \tag{56}$$

where $P(i)_n$ is the value of parameter n at point $\mathbf{P}(i)$ in parameter space. The maximum distance between any two points is given by $\sqrt{12 \cdot 100^2} = 346$ for the case in which

[6]Note that to maintain biological validity, the parameter ranges cannot simply increase without bound.

[7]Note that although a random search case is superior to our algorithm for this set of parameter ranges, we can still make general claims about our algorithm.

every parameter in $\mathbf{P}(a)$ is either a minimum or maximum value and every parameter in $\mathbf{P}(b)$ is the other extreme value. The maximum distance between a point, $\mathbf{P}(a)$, and the canonical set of parameters is given by $\sqrt{12 \cdot 50^2} = 173$ in which $\mathbf{P}(a)$ contains all extreme parameter values.

For the first evaluation of a trial, the initial point in parameter space was determined by picking random values from uniform distributions that were defined by the ranges of each parameter. For the 357 successful trials, the average starting distance from the canonical value was 99 ($\sigma = 13$), the average ending distance from the canonical value was 111 ($\sigma = 16$), and the average distance between the starting and ending values was 105 ($\sigma = 43$).

As a result of the method used to begin each trial, the canonical-to-starting distances represented a uniform distribution throughout parameter space. Because the mean and standard deviation of these distances were approximately equal to the mean and standard deviation of the canonical-to-ending distances, the final bursting points also represented a uniform distribution throughout parameter space. Therefore, the final bursting points were not clustered around the canonical or any other specific value. In addition, the final bursting points were not typically found by making small adjustments to the starting random points. The average starting-to-ending distance of 105 between these values supported this finding. In fact, the 13 trials that were successful on the first evaluation contributed zero distances to this overall distribution, which lowered this mean distance.

These results implied that we found bursting points that represented a wide variety of parameter combinations. For example, Figure 41A shows a plot of the 357 final values of the spiking maximal conductances, $\bar{g}_{\mathrm{Na}}$ and $\bar{g}_{\mathrm{K}}$, which are uniformly distributed throughout the possible pre-defined ranges. We examined all of the parameter combinations in this two-dimensional manner and found that only the spiking activation half maximal voltages, $\theta_q(I_{\mathrm{Na}})$ and $\theta_q(I_{\mathrm{K}})$, did not fully utilize their allowable pre-defined ranges as shown in Figure 41B. The combination of these parameters, not just their absolute values, were noticeably important as illustrated by the distinct region of successful ending points. As

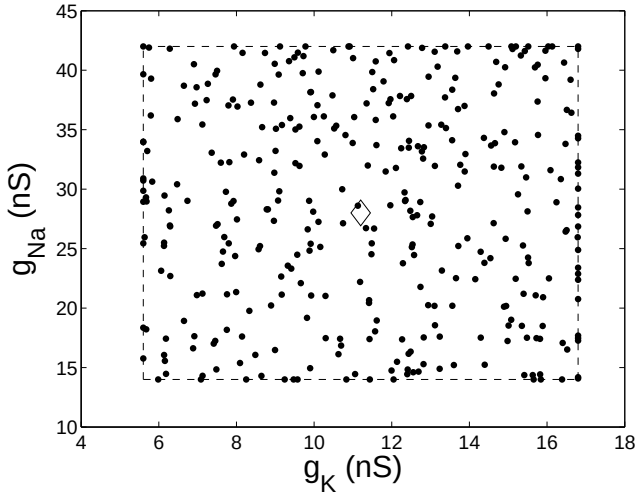
a result, these two activation half maximal voltages, given their pre-defined ranges,[8] were the most sensitive parameters in terms of forcing the neuron into a non-bursting region regardless of the values of the other model parameters. In fact, the roughly uniform distribution of final bursting points and the critical relationship between the spiking activation half maximal voltages were evidence of a complex interplay that existed between the parameters. This parameter interaction was required to obtain a specific type of neural waveform from the underlying nonlinear dynamical system. We will further evaluate the role of these parameters in the next section.

Further Evaluation of the Spiking Activation Half Maximal Voltages. The initial values of $\theta_q(I_{\mathrm{Na}})$ and $\theta_q(I_{\mathrm{K}})$ for the failed trials, as shown by the open circles in Figure 41B, also indicate a definite bias outside of the region of successful bursting points, but an absolute conclusion cannot be made—although the failed trials tended to start outside of this region, starting outside (inside) of the region does not guarantee failure (success).

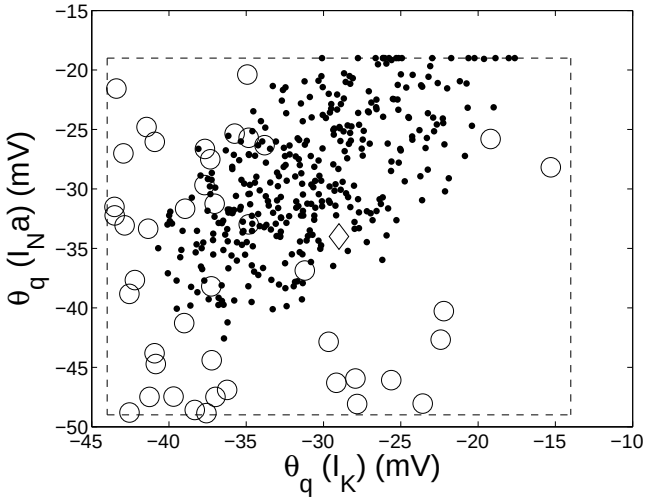
Using the data from the 400 trials for the 120-percent parameter ranges, we further examined the roles of $\theta_q(I_{\mathrm{Na}})$ and $\theta_q(I_{\mathrm{K}})$. Figure 42 shows how the allowable region of values expanded in such a way that it qualitatively retained its general shape. As a result, this region of allowable final values not only depend on the relationship between the two parameters but also depend on their relationships with the other spiking parameters. This result also indicates how the parameter space is mapped—the constrained ranges of the other spiking parameters defines the mapping of the allowable $\theta_q(I_{\mathrm{Na}}) - \theta_q(I_{\mathrm{K}})$ region. In addition, we also believed, but did not prove, that a greater (smaller) concentration of points in some parts of the region corresponded to a larger (smaller) range of allowable values for the other spiking parameters. After evaluating the role that individual parameters had on the success of our algorithm, we examined the results of a single representative trial, which is the topic of the next section.

Evaluating a Single Trial. Using the results from the initial search, a representative trial was chosen to show the progression of two of the maximal conductances ($\bar{g}_{\mathrm{Na}}$ and $\bar{g}_{\mathrm{K}}$).

[8]Note that for smaller ranges for $\theta_q(I_{\mathrm{Na}})$ and $\theta_q(I_{\mathrm{K}})$, the uniform distribution of points depicted in Figure 41A would be more closely matched in Figure 41B.



A.



B.

Figure 41: Parameter-space search algorithm—the 357 final values of selected parameters for the successful trials using the parameter ranges shown in Table 14. The dashed boxes indicate the possible ranges, and the hollow diamonds in the middle of the ranges indicate the canonical values. A. The spiking maximal conductances, $\bar{g}_{\mathrm{Na}}$ and $\bar{g}_{\mathrm{K}}$. B. The spiking activation half maximal voltages, $\theta_q(I_{\mathrm{Na}})$ and $\theta_q(I_{\mathrm{K}})$. The open circles indicate the initial random parameter values for the 43 failed trials.

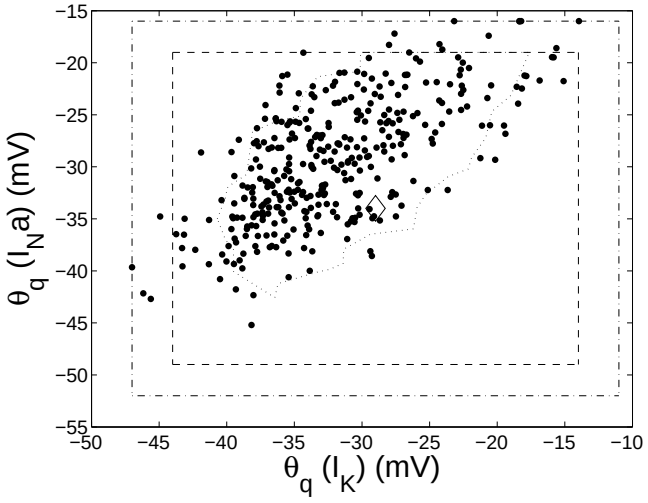


Figure 42: Parameter-space search algorithm—the final values of the spiking activation half maximal voltages, $\theta_q(I_{\mathrm{Na}})$ and $\theta_q(I_{\mathrm{K}})$, for the successful trials using the 120-percent ranges. The smaller dashed box indicates the 100-percent ranges, the larger dashed-dotted box indicates the 120-percent ranges, the hollow diamond in the middle of the ranges indicates the canonical values, and the dotted region inside the boxes indicates the boundary of the final values using the 100-percent ranges.

Figure 43 illustrates that for this particular trial, $\bar{g}_{\mathrm{Na}}$ covered about two-thirds of its range, starting close to its maximum possible value and ending below its published value, and $\bar{g}_{\mathrm{K}}$ covered about one-third of its range, moving between its minimum possible value and just below its published value. These wide-ranging trajectories that were required to locate a successful bursting point for this trial were also indicative of other parameter paths for this and other trials.

The final points of these two maximal conductances were found to be randomly distributed throughout the allowed parameter space as shown in Figure 41A. These final values are clearly not clustered around the canonical points, and in fact, the points that lie on the edges of the dashed region (the minimum and maximum possible values) indicate that these parameter ranges can be increased. For our last experimental results topic, we will discuss the relationship between *all* of the bursting points found using our algorithm.

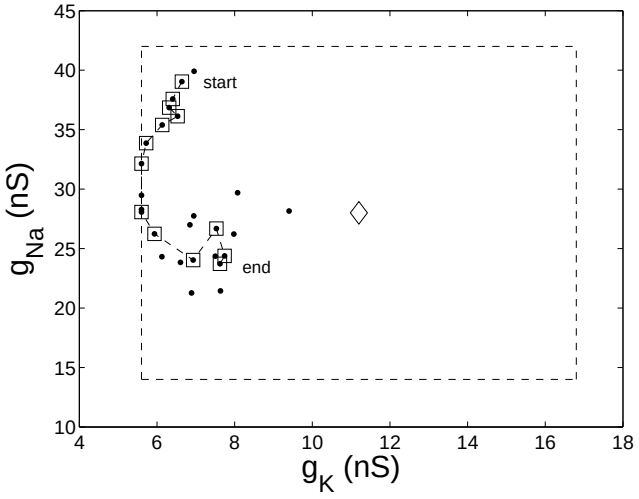


Figure 43: Parameter-space search algorithm—following the progression to successful bursting for a specific trial for the two spiking maximal conductances, $\bar{g}_{\mathrm{K}}$ and $\bar{g}_{\mathrm{Na}}$. The dashed box indicates the possible ranges, and the hollow diamond indicates the published values. The squares indicate the 13 evaluations in which the cost function improved. This trial required 26 evaluations.

Connecting the Bursting Points. We determined the relationship between the 357 bursting points found in the initial 400 trials[9]—that is, whether they were located in numerous "islands" of bursting regions or were "connected" within one contiguous bursting region.

To connect these bursting points, we needed to determine the following: (1) a method to move the parameters and (2) an order to add bursting points to the contiguous region. First, we discretized each of the parameter ranges shown in Table 14 into 60 equivalent bins (this value resulted in a good tradeoff between increment size and computational time). Based on these discrete bins, we used a distance measure, D_{D}, to determine the distances between the bursting points. Note that this distance measure given by Eq. (57) differs from the one given by Eq. (56) because we wanted a measure that directly correlated with the number of changes in parameter values required to connect two points. To determine this

[9]Recall that a trial ended in either success or failure when one of the two following situations occurred: (1) the cost function fell below a threshold ($C_{\mathrm{threshold}} = 100$) or (2) the maximum number of evaluations was obtained ($E_{\max} = 500$).

measure, the distance between any two points, $\mathbf{P}(a)$ and $\mathbf{P}(b)$, is given by:

$$D_{\mathrm{D}} = \sum_{n=1}^{12} \left| P(a)_n - P(b)_n \right| \tag{57}$$

where $P(i)_n$ is the discretized value of parameter n at point $\mathbf{P}(i)$ in parameter space. The maximum distance between any two points is given by $12 \cdot 60 = 720$ for the case in which every parameter in $\mathbf{P}(a)$ is either a minimum or maximum value and every parameter in $\mathbf{P}(b)$ is the other extreme value. The maximum distance between a point, $\mathbf{P}(a)$, and the canonical set of parameters is given by $12 \cdot 30 = 360$ in which $\mathbf{P}(a)$ contains all extreme parameter values.

Using the distance measure given by Eq. (57), we calculated the distances between all 357 bursting points, and we determined the initial connection order of the contiguous bursting region by using the two bursting points that resulted in the minimum distance. We then found the bursting point (from the remaining 355) that gave the smallest distance to one of the two bursting points in the contiguous region and added that point to the connection order. This process continued until a complete order was determined to connect all 357 bursting points. We chose this distance-based connection method, as opposed to a random one, to increase the probability that connection attempts succeeded. A graphical description of the first ten connections is shown in Figure 44.

The connection search began by considering the first two bursting points that initially formed the contiguous region and continued by adding individual bursting points to the region until all 357 bursting points were successfully added. When connecting two bursting points, the parameters to be moved were selected by a random-weighted method—the more increments a particular parameter was required to move between the two bursting points, the more likely it would be chosen. After a single parameter was randomly picked, its value was changed by the defined increment size, a 60-second simulation was recorded, and its cost function was calculated. For this connection search, the minimum cost-function threshold for bursting was set to 250 (*i.e.*, $C_{\mathrm{threshold}} = 250$). We relaxed the value of the threshold because maintaining the original threshold value of 100 was not realistic because many of the 357 bursting points were found with cost function values slightly below 100. We

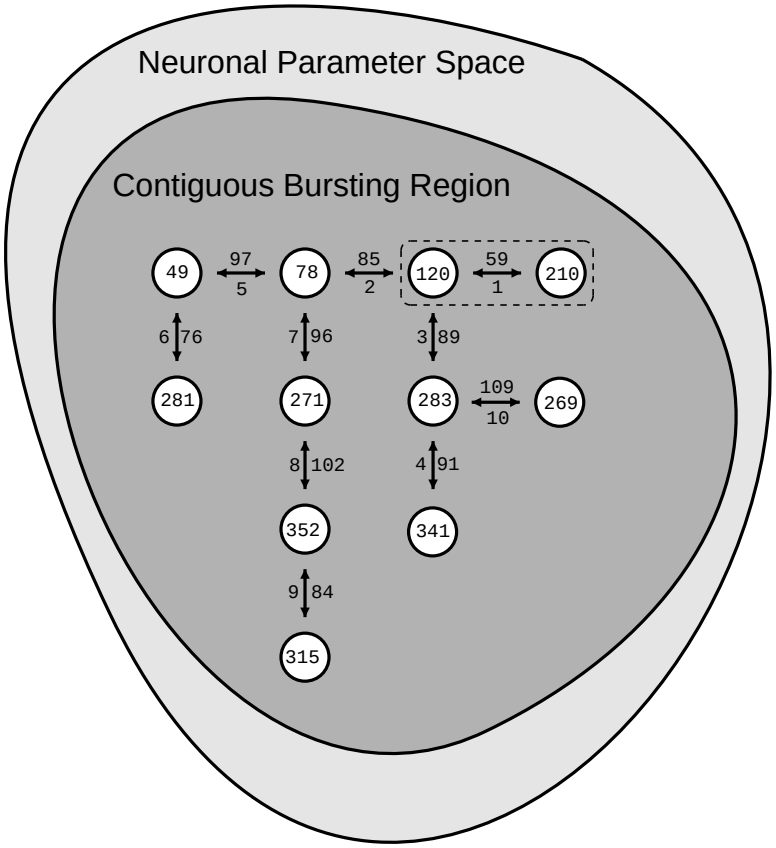


Figure 44: Parameter-space search algorithm—connection diagram for the first ten connections that form the beginning of the contiguous bursting region within the entire neuronal parameter space. The numbers in the circles represent one of the 357 bursting points. The numbers to the right and above the arrows are the distances, D_{D}, between the bursting points. The numbers to the left and below the arrows are the order in which the connections were made. The connection in the dashed horizontal box (upper right) is the shortest distance between any two bursting points and sets the initial connection order that forms the contiguous bursting region.

still maintain, however, that the new threshold was still a conservative value for classifying bursting. If bursting was achieved for the current parameter move, then the probability of picking that parameter for the next parameter move was marginally smaller (until it became 0 to preclude further moves). A trial ended in either success or failure when one of the two following situations occurred: (1) all of the parameter moves were successfully completed or (2) more parameter moves were required but none resulted in bursting. If a trial succeeded, the bursting point was added to the contiguous region and the next bursting point, as determined by the connection order previously found, was used in the next trial. If a trial failed, the next most recently added bursting point to the contiguous region was used to attempt to make the connection.

Applying our connection methods and constraints, we determined that a single contiguous bursting region included all of the 357 bursting points found in our initial experiment. We were confident that we were not "skipping over" a non-bursting region because of the sufficiently small parameter increment size that was used. Given the high-order dimensionality of the parameter space, we could not determine the shape or size of this contiguous bursting region. We were also confident that the cost-function threshold could be strengthened if more valid bursting points were added to our program. The existence of this contiguous bursting region implied that these parameter sets that produce similar spectral content were not random combinations, but were rather closely related and, in fact, connected in parameter space. This concludes the analysis of our experiments, and we follow with our closing comments.

3.1.4 Improvements to the Algorithm

We considered improvements to our search algorithm because better, more-efficient algorithms can be implemented, but we also know that the *no free lunch* theorems [132] indicate that no algorithm is necessarily the best one for all cases. The standard tradeoffs—accuracy versus computational time—must be considered when making improvements, which are described in the following paragraphs: (1) frequency analysis, (2) cost function and the bursting classification, and (3) steepest-descent approach.

Frequency Analysis. An analysis of the frequency data will provide more information than a simple reduction to three frequency bins followed by a summation of those bins. For example, the *spikiness* of the data can be measured by comparing the mean and maximum energy values. A simple method is to sum the frequency data using a *boxcar* approach; a frequency component that moves across a boundary for a change in model parameters, however, will have a larger impact on the cost function than if an alternative *windowing* method were employed. The definitions of the frequency bins were loosely determined by the target burst spike frequency, but more and different frequency bins can be used. Another method is to vary the definitions of the bins such that the sum of the powers in each bin is equivalent. Also, the arguments of the `spectrum` command in MATLAB can be adjusted to optimize the results.

Cost Function and the Bursting Classification. Calculating the squared error difference is a common technique used in optimization routines, but a quartic difference in our cost function will provide a smoother descent to the goal. The low-, mid-, and high-frequency components of the cost function are weighted equally, but the cost function should not disproportionately penalize an output that has too much or too little low-frequency content because it primarily determines the period and duty cycle of the bursting (*i.e.*, the cost function should be duty-cycle invariant). By construction, our cost function is setup to find a specific bursting characteristic and therefore results in an extremely conservative classification scheme. As a result, a biologically realistic bursting output with frequency components much different than those given by the target output will not meet the cost-function threshold for our definition of bursting, and the search will be artificially extended. The size of the bursting regions that we are locating are set by our arbitrary and specific definition of bursting. Therefore, a variety of definitions of bursting could be used. Rather than using a simple threshold to determine a specific type of bursting, a more complex definition of bursting will shorten the time required to find a general bursting output and thus expand the size of the desired bursting region. This alternative, however, provides an example of an ill-defined classification problem—that is, achieving the simultaneous goals of broadening the outputs classified as bursting and minimizing the mis-classified bursting

outputs.

Steepest Descent Approach. A true gradient descent requires information from incrementally moving each of the 12 parameters individually, but this approach will substantially increase the testing time. In addition, our process uses derivatives, not acquired knowledge of the system. For example, if the current point in parameter space did not result in bursting, then the next set of points was in the opposite direction from the previous move (plus some random variation). An alternative approach could determine if the current cost function value was close to the smallest previous cost function value, and if so, then the next point would be in the vicinity of the current point. Our method and the alternative approach are both valid steepest descent-type methods, but because we were satisfied with the results obtained using the mathematical method first, the alternative method was not necessary.

3.2 A Comparative Analysis of Multi-Conductance Neuronal Models in Silico

We demonstrated that the flexible silicon-neuron architecture [106] presented in Section 2.2 can implement three disparate conductance-based neuron models with both fast and slow dynamics.[10] By exploiting the real-time nature of this physical implementation, we mapped the model dynamics across a large region of parameter space. We also found that two of these dynamically different models represent points in a contiguous bursting space that spans between the two models. By systematically varying the model parameters, we also found that multiple, diverse trajectories in parameter space connected the two canonical bursting points. In addition, we found that the *combination* of parameter values kept the neuron in the bursting region. These findings demonstrated the usefulness of the silicon-neuron architecture as a neural-modeling tool and illustrated its versatility as a platform for a multi-behavioral neuron that resembles its living analog. The framework of this section is as follows: In Section 3.2.1, we will describe the features, relationships, and differences between three neural models, including implementation details specific to our tests and that

[10]The material in this section was taken from a paper by DeWeerth, Reid, Brown, and Butera that was accepted by Biological Cybernetics[22].

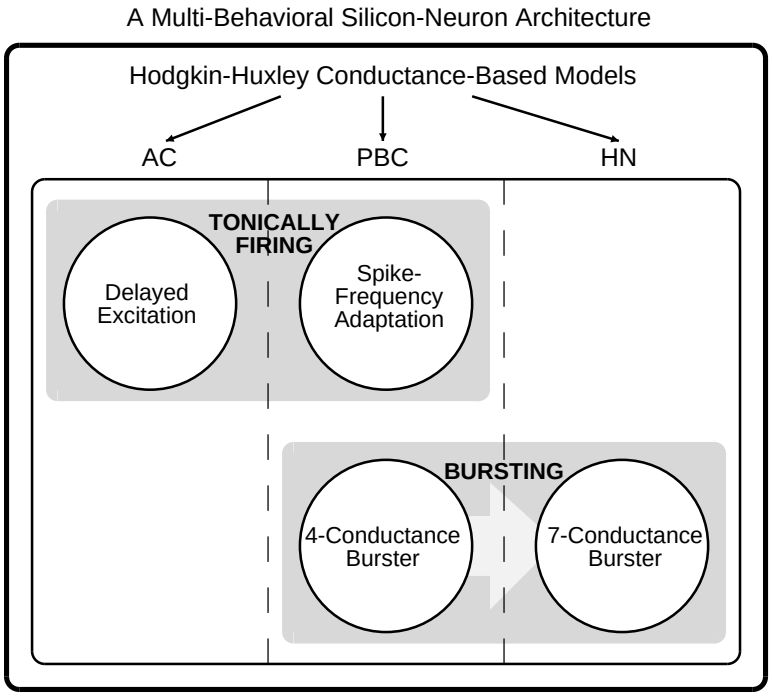


Figure 45: Comparative analysis—neuronal model overview of the salient features that we examined.

were not covered in Section 2.2. In Section 3.2.2, we will show our experimental results, including three different ways that we spanned the bursting space. In Section 3.2.3, we will give our interpretations of the results.

3.2.1 The Neural Models—Features, Relationships, and Differences

The three neural models used in this study will be referred to as the HN, AC, and PBC models. Figure 45 is a graphical representation of the relationships between these models and their salient functional behaviors. The following describes our rationale for choosing these models and some of their important features:

HN Model. A model of the heartbeat interneuron of the medicinal leech [86] [51] was the basis for the silicon-neuron architecture used in this study. This silicon neuron was also used to study rhythmic motor-pattern generation in a hybrid system, in which electronic circuits were interfaced through a dynamic clamp to living neurons [107]. The model contains six active conductances including non-activating and hyperpolarization-activated currents and produces a rich assortment of voltage-dependent activity modes including bursting, tonic

firing, and silence.

AC Model. A classical A-current model [14] of a swim interneuron was chosen because it generates qualitatively different spiking dynamics than that of the bursting HN model. The AC model uses a fast, transient potassium current to explain the long latency of the first spike (delayed excitation) [37]. Its most recent improvement [16] was the basis for the set of parameters used with our implementation.

PBC Model. A model (Section 2.1.1) of an oscillatory bursting neuron involved in the respiratory rhythm generation in the PBC in mammals [8] provides a link to the other models. The PBC model generates complex dynamical behaviors similar to those of the HN model, but with fewer conductances (reduced model neuron). In addition, this model was used to demonstrate spike-frequency adaptation, a different tonically firing dynamical response than with the AC model.

These three conductance-based models use fast ($1 - 10$ ms) and slow ($1 - 5$ s) time constants and voltage-dependent processes to produce their complex dynamical behaviors. Both the HN and PBC models produce oscillatory bursting behaviors, which was essential to our study of central pattern generators [79]. Therefore, a focus of this study was on the regions of parameter space where bursting occurred in these two models.

The HN and PBC models also differ in the following features: (1) neuron functional type (leech heartbeat interneuron versus respiratory rhythm generation neuron), (2) number of model parameters (29 versus 16), (3) conductance types and number (six active conductances including a hyperpolarization-activated current versus three active conductances), and (4) characteristics of the burst dynamics (shown later in Figure 50).

Upon considering these similarities and differences and evaluating the dynamics of the two bursting models, we studied the parameter spaces where oscillatory bursting occurs in these two models. A qualitative examination of these results suggested that the bursting regions of these two models shared a contiguous region of parameter space as indicated in Figure 46. This analysis complemented our study of the flexibility of the silicon-neuron architecture.

This specific implementation of the silicon-neuron architecture was designed by Simoni

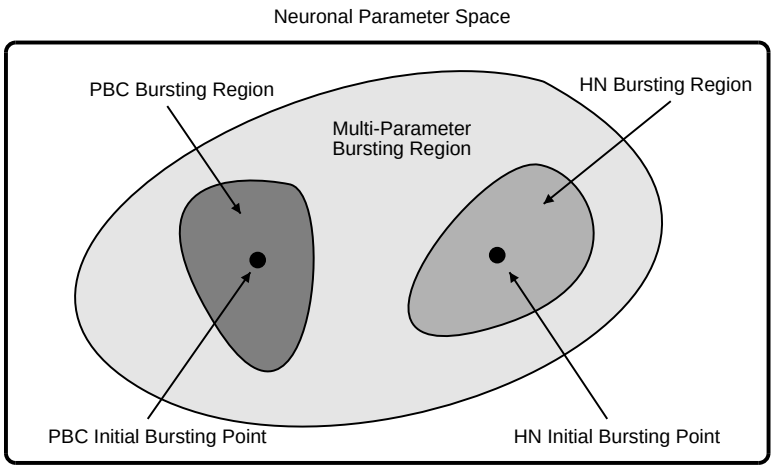


Figure 46: Comparative analysis—the hypothesized relationship between the PBC and HN bursting regions.

[106] to produce neural activity using a HH formalism [55] in which it emulated several currents contained in a previously published model of an oscillator heart interneuron [51]. The description of the silicon-neuron architecture, including the approximations that were made, was given in Section 2.2.1. We implemented the three models using the following transmembrane current equations that require a variety of the ionic currents:

$$C_{\mathrm{mem}}\frac{dV_{\mathrm{mem}}}{dt} = I_{\mathrm{Leak}} + I_{\mathrm{Na}} + \begin{cases} I_{\mathrm{P}} + I_{\mathrm{K1}} + I_{\mathrm{K2}} + I_{\mathrm{Ca}} + I_{\mathrm{h}} \text{ (HN model)}, \\ I_{\mathrm{K}} + I_{\mathrm{NaP}} \text{ (PBC model)}, \\ I_{\mathrm{K}} + I_{\mathrm{A1}} + I_{\mathrm{A2}} \text{ (AC model)} \end{cases} \tag{58}$$

where C_{mem} is the whole cell capacitance, V_{mem} is the membrane potential (the neuronal output), and t is time. The middle equation is the same as Eq. (1), without the extrinsic currents. The multi-conductance models comprise a wide variety of voltage- and time-dependent currents [52] including a passive leakage current (I_{Leak}), sodium currents (I_{Na}, I_{P}, and I_{NaP}), potassium currents (I_{K}, I_{K1}, and I_{K2}), a calcium current (I_{Ca}), a hyperpolarization-activated current (I_{h}), and an A-current (I_{A1} and I_{A2} make the independent activation terms explicit). The description of the HH dynamics of the gating variables was given in Section 2.1.1, and a complete mathematical and circuit description of this version of the silicon neuron was previously published [106]. Table 15 shows the conductance blocks used in the silicon model neuron to implement each model. Only the maximal

Table 15: Comparative analysis—conductance blocks used in each model (number of parameters in parenthesis).

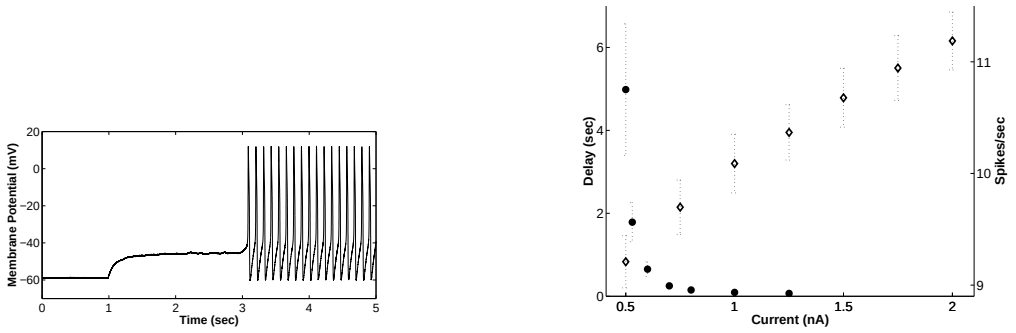
Description	HN	PBC	AC
passive leak current (2)	I_{Leak}	I_{Leak}	I_{Leak}
fast sodium current (5)	I_{Na}	I_{Na}	I_{Na}
persistent sodium current (3)	I_{P}	not used	not used
inactivating potassium current (6)	I_{K1}	not used	I_{A2}
slow, non-inactivating potassium current (3)	I_{K2}	I_{K}	I_{K}
slowly inactivating low-threshold calcium current (6)	I_{Ca}	I_{NaP}	I_{A1}
hyperpolarization-activated inward current (4)	I_{h}	not used	not used

conductances of the I_{Leak}, I_{Na}, I_{K2}, and I_{Ca} blocks were set to nonzero values for all three models. The voltage of the maximal conductance parameter can effectively shut off the entire block when it is not being used (*i.e.*, $\bar{g}_i = 0$). The number in parenthesis indicates the number of voltage biases that need to be set for that specific ionic channel (a maximum of 29 are required).

3.2.2 Results

We first demonstrated the flexibility of a single silicon-neuron architecture by implementing the three disparate neuronal models and by examining their resulting multi-behavioral functionality (delayed excitation, spike-frequency adaptation, and bursting activity). The AC and PBC models were used to demonstrate different responses from tonically firing outputs. We then explored and related large regions of parameter space and showed that the bursting regions of the two bursting models, HN and PBC, are enclosed within a single multi-parameter bursting region and that multiple, diverse paths connect the two canonical bursting points.

Model Implementations. We performed a series of tests to demonstrate different aspects of the neuronal models, thereby illustrating the versatility of the silicon-neuron architecture and giving an indication of its potential as a neural modeling tool. Employing both fast ($1-10$ ms) and slow ($1-5$ s) time constants, the neurons produced a variety of qualitatively different outputs. In addition, we will give an indication of why the integrated-circuit chip functions differently from the computer model because of the circuit approximations discussed in Section 2.2.1.



A. B.

Figure 47: Comparative analysis—repetitive firing properties from the AC model. A. Representative spike train evoked by a 0.5-nA current pulse. The first spike is delayed by 2.1 s from the onset of the current pulse at 1.0 s. B. Plots of spike frequency (open diamonds) and latency of the first spike (filled circles) as a function of injected current. The dashed vertical error bar lines represent one standard deviation above and below the mean.

Test 1. The goal of this test was to reproduce the firing properties, in terms of delay and spike frequency in response to a current pulse, as published in the literature for an AC model [14]. The AC model was implemented using leak, Na^+, K^+, and A currents with two independent activation terms. We expected to see a long first spike latency (also referred to as the delay, or utilization time) following a current stimulus, and Figure 47A verifies this finding with the integrated-circuit chip. This latency is explained by the A-current, a fast, transient, potassium current that activates on depolarization [14]. In addition, we expected two results from the delay and spike frequency measurements for a series of current pulses: (1) An inverse hyperbolic relationship between the magnitude of the current pulse and the latency of the first spike and (2) A linear relationship between the magnitude of the current pulse and the spike frequency. Figure 47B illustrates these relationships with the chip. Both of the figures in Figure 47 are qualitatively similar to figures published in the original A-current literature [14].

Test 2. The goal of this test was to demonstrate how the instantaneous spike frequency of an endogenous burster, located in a silent region of parameter space, changes in response to a series of current pulses. To perform this test, we initialized the PBC-configured neuron to its canonical[11] bursting point and reduced the maximal conductance of the burst-generating

[11]For the purposes of this study, a *canonical point* simply locates the neuron in a bursting region and is

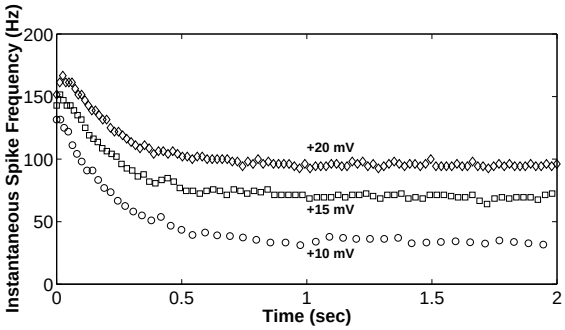


Figure 48: Comparative analysis—spike-frequency adaptation in the PBC model for the three different step changes in E_{Leak} indicated in the plot ($E_{\text{Leak}} = -60$ mV at 0 s).

current, $\bar{g}_{\text{NaP}}$, to locate the neuron at a point in parameter space in which bursting does not occur. We used a perturbation in E_{Leak} as a proxy for an externally applied stimulus current[12]; an increase in E_{Leak} (depolarization) is equivalent to a positive applied current injected into the output node of the neuron, and in this case, an adapting response will result due to the slower kinetics of the recovery variable. We performed three tests in which E_{Leak} was increasingly depolarized from its starting value of -60 mV ($\Delta E_{\text{Leak}} = 10$ mV, 15 mV, 20 mV). The resulting spike-frequency adaptation is illustrated in Figure 48 in which the initial large spike frequency quickly settles to a lower steady-state level. The data show the following: (1) the steady-state values were achieved at approximately one second and (2) for larger changes in E_{Leak} (more depolarization), the larger the steady-state frequency and the smaller the relative decrease from the initial spike frequency.

Test 3. Computer simulations of the PBC model were used to compare the differences in the bursting regions between the fully implemented published model [8] and one that reflects the silicon-neuron implementation with the simplifications described in Section 2.2.1. A comparison of the size of the bursting region in the $E_{\text{Leak}} - \bar{g}_{\text{NaP}}$ space for the two implementations is shown in Figure 49A and Figure 49B. Although the bursting region is

not necessarily tied to the values published in the literature.

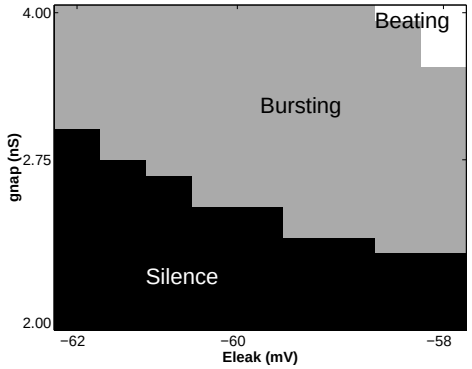
[12]Note that a change in E_{Leak} corresponds to a constant change in the leak current (proportional to $\bar{g}_{\text{Leak}}$), and therefore, a step change in E_{Leak} maps directly to a current stimulus.

noticeably larger for the fully implemented version, the outputs for these two implementations shown in Figure 49C indicate that the macro bursting dynamics at the canonical point are qualitatively similar. Consequently, we believe that the circuit-design decisions did not impede our ability to reproduce neurophysiological behaviors with the physical implementation.

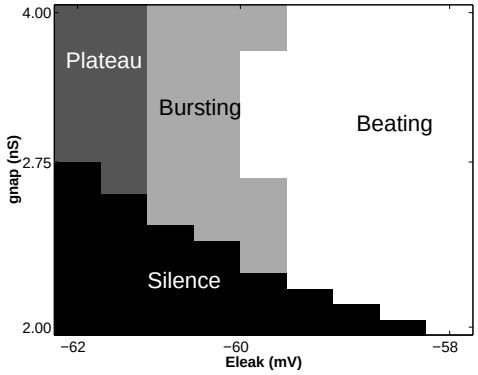
Studying the Parameter Space. Canonical bursting points were found for the two bursting neuron models, the PBC and HN models, and their representative outputs are shown in Figure 50. Our initial analysis of the dynamics of the HN and PBC models suggested that their bursting regions are related. To test this hypothesis, we attempted to "connect" these models in parameter space. We first set the initial point of the test to be the canonical bursting point of one of the models. While ensuring that bursting activity is maintained, we then moved parameters individually until the parameter values corresponded to the canonical bursting point of the other model. To accomplish this aim, we needed to determine the following: (1) By what quantity should the parameters by moved (*i.e.*, the parameter step size)? (2) How will the different paths traversed between the two canonical bursting points in parameter space be quantitatively differentiated? (3) What methods will be used to determine these different paths?

Without a "textbook" definition of bursting, we were also required to establish a set of rules to indicate whether neural activity moved outside of the bursting region (*i.e.*, to determine where bursting exists and is stable). This procedure illustrated an example of an ill-defined classification problem—that is, for a given set of parameters, the transition between the end of one bursting region and the beginning of another activity region must be determined unambiguously, although the definition of these boundaries is subjective.

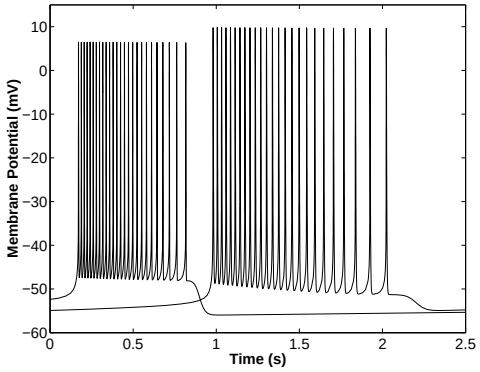
The silicon-neuron parameters were initially set to those that implemented the PBC model. Consequently, the half-maximal voltages, reversal potentials, and time constants of the three conductances that were shut off in the PBC model were set to their proper values in the HN model, reducing the number of parameters that needed to be moved. In addition, the leak reversal potential, E_{Leak}, was set to -60 mV for both models and provided



A.

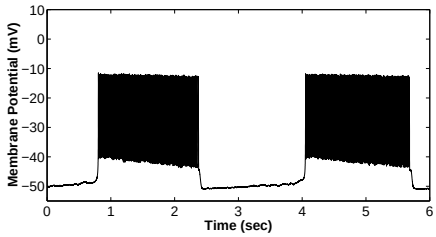


B.

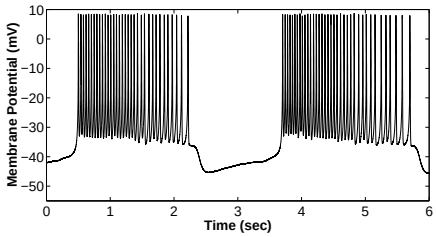


C.

Figure 49: Comparative analysis—SIMULINK simulation results for the PBC model. The bifurcation diagrams show the intrinsic modes of single-cell activity as a function of $\bar{g}_{\mathrm{NaP}}$ and E_{Leak} for A. the fully implemented model and B. the version implemented in silicon. C. A comparison of the outputs at the canonical point with the data from fully implemented model on the left side of the figure.



A.



B.

Figure 50: Comparative analysis—representative outputs of the two bursting HH models. The spike heights, baseline membrane potentials, and average spike frequencies are much different between the two models but nonetheless both models exhibit bursting behavior. A. PBC model (average spike frequency $\approx$ 85 Hz, spikes per burst $\approx$ 150). B. HN model (average spike frequency $\approx$ 20 Hz, spikes per burst $\approx$ 35).

the reference for all model voltages.[13] Therefore, to complete the transition between the bursting models, only 19 of the 29 HN parameters needed to be moved between the starting values of the PBC model and the ending values of the HN model. We moved each of the 19 parameters using 30 equi-voltage increments for a total of 570 moves, or trials. We chose 30 equal increments as a good compromise between larger parameter step sizes that forced the model irrevocably outside of its bursting region and smaller parameter step sizes that unnecessarily added to the already lengthy testing time. In fact, small step sizes were limited by the digital resolution of the input devices.

After obtaining baseline bursting activity from the PBC model, the first of the 570 trials began by randomly choosing one of the 19 parameters, moving its value one-thirtieth of the distance between its starting and ending values, recording 40 seconds of data at 4000 samples/second, and conducting a time-based analysis. Outputs of subsequent trials were

[13]If all half-maximal voltages and reversal potentials were moved by the same quantity, then the output would not change. Therefore, one of these voltages can be (arbitrarily) defined as the reference and be set to a constant value. We chose E_{Leak} to be this voltage.

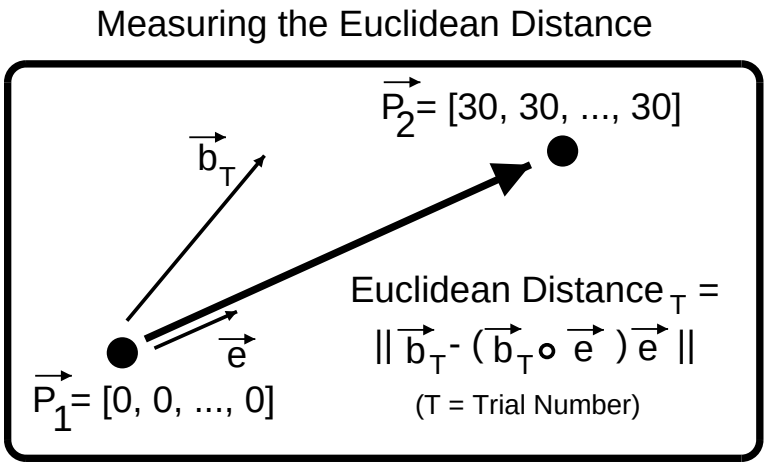


Figure 51: Comparative analysis—the measurement of the Euclidean distance in a discretized multi-parameter bursting space.

considered to be bursting[14] if the following occurred: (1) the average spiking and silent times were within 60 percent of the previous successful burst, (2) the average number of spikes per burst and average spike frequency were no more than 120 percent greater than those of the previous successful burst (no constraint was placed on smaller values), (3) the standard deviation of the spike heights was less than one percent of the mean (to avoid the degenerate bursting regions), and (4) the number of data points above the spiking threshold (10 mV below the mean spike heights) was at least one percent (to avoid doublet- and triplet-type bursting that can occur with a non-bursting mechanism).

These were considered to be conservative thresholds because many times a qualitatively good bursting output was not considered successful because one or more of the criteria was not met. If all of the bursting criteria above were met (*success*), then the parameter remained at its new value; otherwise, the parameter was moved back to its previous value (*failure*). Then, another parameter was chosen (depending on one of the three methods as explained below), and the process was repeated. The test concluded when the 570 trials were deemed successful.

We used a Euclidean distance measure to differentiate the different paths taken between

[14]Bursting is qualitatively defined as periodic repetitive activity punctuated by periodic episodes of silence. The burst cycle consists of an active phase, characterized by the repetitive firing of action potentials (spiking), and a silent phase, characterized by a membrane potential that varies slowly at a hyperpolarized value (silent).

the canonical bursting points of the two models in the discretized multi-parameter bursting space. The calculation of this distance is illustrated in Figure 51. The initial and final parameter vectors are defined by $\overrightarrow{P_1} = [0, 0, \ldots, 0]$ and $\overrightarrow{P_2} = [30, 30, \ldots, 30]$, respectively, where $\overrightarrow{P_1}$ represents the discretized canonical bursting point for the PBC model and $\overrightarrow{P_2}$ represents the discretized canonical bursting point for the HN model. $\overrightarrow{e}$ defines a unit vector along the direct path between $\overrightarrow{P_1}$ and $\overrightarrow{P_2}$, in which the direct path is defined by a vector with equal components; $\overrightarrow{b}_T$ is the actual vector between $\overrightarrow{P_1}$ and the current point after T trials. These vectors are represented by the following:

$$\overrightarrow{e} = \frac{1}{\sqrt{p}}[1, 1, \ldots, 1] \tag{59a}$$

$$\overrightarrow{b}_T = [p_1, p_2, \ldots, p_{19}] \tag{59b}$$

where p is the number of parameters that needs to be varied (19 for our test) and p_i represents the discretized value of parameter i for $i \in \{1, 2, ..., p\}$. The Euclidean distance after T trials, d_T, as measured by the difference between $\overrightarrow{b}_T$ and the projection of $\overrightarrow{b}_T$ on the direct-path vector (defined by $\overrightarrow{e}$), is calculated as:

$$d_T = ||\overrightarrow{b}_T - (\overrightarrow{b}_T \circ \overrightarrow{e})\overrightarrow{e}|| = ||\overrightarrow{b}_T - \overrightarrow{v}_T|| \tag{60}$$

where $\overrightarrow{v}_T$ is the projection of $\overrightarrow{b}_T$ onto $\overrightarrow{e}$.

Our method of exploring the parameter space of a conductance-based model to produce a desired behavior is in contrast to a previous study in which a database of approximately 1.7 million single-compartment model neurons were generated by varying maximal conductances based on measurements from lobster stomatogastric neurons [94]. Evaluating our 19-parameter space (that includes all model parameters as opposed to maximal conductances only) in a similar manner would not be feasible. Also, by its nature, the physical system works in real-time and would not benefit from the additional computing power that was used in their study. Instead, we used automated smart search algorithms to obtain our results.

We used three different methods to move between the canonical bursting points: (1) uniform parameter variation, (2) non-uniform parameter variation, and (3) random parameter

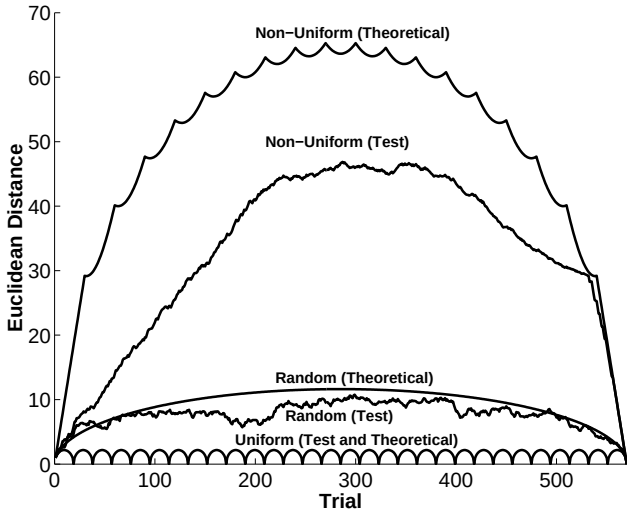


Figure 52: Comparative analysis—Euclidean distances as the parameters were individually moved in discrete steps from one neural bursting model to the other. The test and theoretical distances are shown for three parameter variation cases: uniform, non-uniform, and random. Note that the test and theoretical distances for the uniform case are indistinguishable.

variation. The distances calculated by the first two methods provided boundary conditions. Note that the trial number, T, was chosen for the x-axis of the distance plots because it is proportional to the projection of $\overrightarrow{b}_T$ along $\overrightarrow{e}$; that is, $T = \sqrt{p} \cdot (\overrightarrow{b}_T \circ \overrightarrow{e})$ because $\overrightarrow{b}_T \circ \overrightarrow{e} = \Sigma p_i / \sqrt{p}$ and $\Sigma p_i = T$. These three contrasting methods were chosen to demonstrate that the bursting models not only share the same bursting region, but also connect in diverse ways. By using a Euclidean distance measure, we were able to quantify the size of this discretized multi-parameter bursting region, thereby allowing us to report the results in a meaningful way. A plot of the theoretical and actual distances for the three methods are shown in Figure 52 and will be explained more fully below. Although we expected the test results to be in the "ballpark" of the theoretical results (which is our reason for showing them in the figure), these results were certainly not assured because we only performed each test one time and random processes are not guaranteed to track to the mean. Note that the goal, however, was not to show how close the test results tracked the theoretical results; instead, the goal was to show successful connections using different defined paths.

Uniform Parameter Variation. For uniform parameter variation, the parameters were

selected in such a way to maintain uniformity across the parameters (*i.e.*, the correlation between the parameter that just moved successfully and the selection of the next parameter is a minimum). This method set the lower bound on the possible distances for the progression from one model to the other. For example, in the theoretical case,[15] $\overrightarrow{b}_T$ follows the following path: $\overrightarrow{b}_1 = [1, 0, \ldots, 0]$, $\overrightarrow{b}_2 = [1, 1, 0, \ldots, 0]$, $\overrightarrow{b}_3 = [1, 1, 1, 0, \ldots, 0]$, $\ldots$, $\overrightarrow{b}_{19} = [1, 1, \ldots, 1]$, $\ldots$, $\overrightarrow{b}_{38} = [2, 2, \ldots, 2]$, $\ldots$, $\overrightarrow{b}_{551} = [29, \ 29, \ldots, 29]$, $\ldots$, $\overrightarrow{b}_{570} = [30, 30, \ldots, 30]$. This method minimizes the distance, d_T, and becomes equal to zero every p trials when

$$\overrightarrow{b}_T = \frac{T}{p}[1, 1, \ldots, 1] = \frac{T}{\sqrt{p}}\overrightarrow{e} \tag{61}$$

where T/p is an integer is satisfied. In this zero-distance case, $\overrightarrow{b}_T$ is along the projection defined by $\overrightarrow{e}$ as shown in Figure 51. As depicted in Figure 52, the test results for this case are indistinguishable from the theoretical distance and imply that the neuron never moved to a non-bursting region; these results also imply that in no instance was $p_i > p_j + 1$ for any two parameters i and j. This result also shows the surprising consequence that the average parameter values of the two canonical bursting points (*i.e.*, $\overrightarrow{P_{avg}} = (1/2)(\overrightarrow{P_1} + \overrightarrow{P_2})$) is also a bursting point. In fact, the path determined by this method produced the most robust of the three methods because every parameter move generated valid bursting as determined by the criteria specified above.

Non-uniform Parameter Variation. For non-uniform parameter variation, the parameters were selected in such a way to maintain non-uniform changes across the parameters (*i.e.*, the correlation between the parameter that just moved successfully and the selection of the next parameter is a maximum, until the parameter has moved to its final destination). This method set the upper bound on the possible distances for the progression from one model to the other. For example, in the theoretical case (*i.e.*, no failures), $\overrightarrow{b}_T$ follows the following path: $\overrightarrow{b}_1 = [1, 0, \ldots, 0]$, $\overrightarrow{b}_2 = [2, 0, \ldots, 0]$, $\overrightarrow{b}_3 = [3, 0, \ldots, 0]$, $\ldots$, $\overrightarrow{b}_{30} = [30, 0, \ldots, 0]$, $\ldots$, $\overrightarrow{b}_{60} = [30, 30, 0, \ldots, 0]$, $\ldots$, $\overrightarrow{b}_{540} = [30, 30, \ldots, 30, 0]$, $\ldots$, $\overrightarrow{b}_{570} = [30, 30, \ldots, 30]$. Figure 52 shows that the notches in the plot of the theoretical distance for this method occur every 30 trials at the point where one parameter has moved to its final value (*i.e.*,

[15] The theoretical, or ideal, case assumes that every move of a parameter results in successful bursting.

30 increments). In the test case, we were unable to use this theoretical path to successfully move between the two canonical bursting points. This result suggests that the complex interplay between the neural parameters requires coordinated parameter movement to maintain bursting; alternatively, a disproportionate number of moves in one or more of the parameters will obviate bursting activity. Instead, we randomly picked the parameters by weighting the selections by the previous results (*i.e.*, the more times a parameter was previously moved, the greater its chance of being selected) to push the upper limits on the distance. This modified non-uniform parameter variation method still resulted in a large number of parameter moves that did not result in bursting (127), suggesting that we were on the border of moving the parameters too far without compensatory movement by other parameters. The distance between the theoretical and test distances shown in Figure 52 indicates the extent of the parameter space in which bursting is not possible, but the test distance still reveals that the maximum distance obtained by this method is more than ten times the distance obtained by the previous method for picking parameters.

Random Parameter Variation. For random parameter variation, the parameters were randomly selected with decreasing likelihood after each selection. An analogous situation is one in which balls are picked from an urn without replacement. The urn contains 30 of each type of $p = 19$ different balls resulting in $570!/(19 \cdot 30!)$ possible paths with differing probabilities (*i.e.*, called the *urn problem*); this can be characterized by the hypergeometric distribution [125]. After substituting the vectors from Eq. (59b) into Eq. (60), we calculated the expected distance squared after T trials, $\mathrm{E}[d_T^2]$, to be the following:

$$\mathrm{E}[d_T^2] = p(\sigma^2 + \mu^2) - T \cdot \mu \tag{62}$$

where μ and σ are the mean and standard deviation. The symmetry of the problem helped to reduce the final form of $\mathrm{E}[d_T^2]$ to a relatively simple expression as shown by Eq. (62). The parameter characteristics μ and σ for this distribution are given by the following:

$$\mu = \frac{T}{p} \tag{63a}$$

$$\sigma = \frac{T \cdot m \cdot n \cdot (n + m - T)}{(n + m - 1)(n + m)^2} \tag{63b}$$

where n and m are the number of "successes" and "failures", respectively, for selecting a particular parameter; in our case, $n = 30$ for the 30 increments required for a particular parameter and $m = n(p-1) = 540$ (*i.e.*, $m = \max[T] - n$). Note that $\sqrt{\mathrm{E}[d_T^2]} \approx \mathrm{E}[d_T]$, and in fact, $\sqrt{\mathrm{E}[d_T^2]} \geq \mathrm{E}[d_T]$. Therefore, Eq. (62) provides an upper bound on the expected distance, $\mathrm{E}[d_T]$, for this third method. We also verified this closed-form solution by simulating this selection method in MATLAB 10,000 times. In the test case, relatively few (21) parameter moves failed to result in bursting which not surprisingly falls between the uniform and non-uniform methods. The test distance trajectory for this method shown in Figure 52 tracks the expected theoretical distance trajectory fairly well and would have tracked closer if less failures occurred along the way.

3.2.3 Interpretations of the Results

The wide region of bursting activity as shown by the Euclidean distance measures shown in Figure 52 indicate that this region can be specified by a myriad of combinations of parameters. We also showed how the bursting characteristics and parameter changes evolve as the parameters are moved. Figure 53 illustrates, for the three methods used to connect the canonical bursting points, the progression of the major characteristics of the bursting waveforms: silent time, spiking time, period, duty cycle and average spike frequency. Note that the neuron stays in a bursting region for each of the 570 successful trials shown in Figure 53. Although the starting and ending bursting characteristics are approximately equal (by its very nature, the random components within the real, physical system will yield slightly different results), the variability of these bursting characteristics is another illustration of the diverse paths that were taken. Figure 54 reveals the asymmetrical progressions of the maximal conductances for the three methods used to connect the canonical bursting points. The maximal conductances were normalized to their final values because some of these values were an order of magnitude larger than others. Also, note that some of these values decrease to their final values and some increase. This figure also offers another perspective for the paths used to connect the canonical bursting points—that is, the *combination* of parameter values, not a *single* one, maintains the neuron in the bursting region (we did

find, however, that in terms of which parameter moves caused the most failures, the half maximals were seemingly the most sensitive by this measure and the reversal potentials were the least sensitive). This finding is analogous to previously published results that also show that different combinations of conductances produce similar neuronal output behavior, and therefore, no single current individually determines the firing properties of the neuron. This finding suggests two outcomes: (1) The pattern of activity is the goal (*i.e.*, bursting) and is produced by somewhat variable underlying mechanisms (*i.e.*, the model parameters) that affect many interacting nonlinear processes [80]. (2) The effect of modulating a single conductance on the neuronal behavior depends on the neuron (or the location in parameter space) [33].

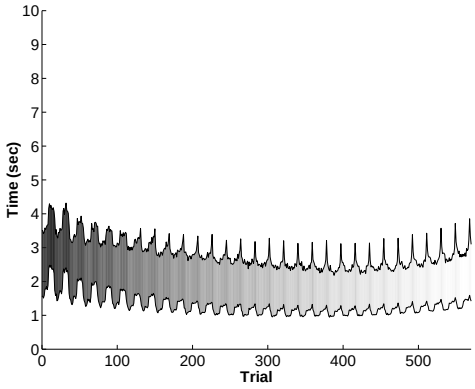
An interesting result we found using the uniform method to connect the two bursting points was that bursting occurred using the average parameter values of the two canonical bursting points, but we do not claim that this is a general result. This finding must be carefully considered and should not be confused with previously published results that determined that setting parameters to the average value of parameter ranges that produced bursting for a *single* neuron did not result in bursting. In this case, the fundamental reason for this failure of averaging (*i.e.*, the *averaged model*) is that the bursting region of the single neuron occupies a concave region of parameter space [40].

The existence of a single contiguous bursting region that contains points representing the HN and PBC models indicates a closer association between these models than previously considered. This concept, demonstrated with an engineered system, ties back to the opening ideas that we presented in this section—that is, living neurons with similar flexible architectures can be used in a wide range of behaviors.

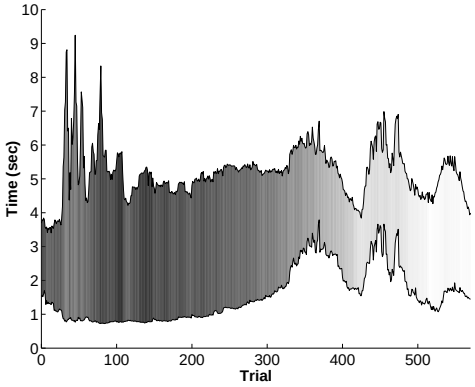
3.3 An Analysis of the Effects of Intrinsic Heterogeneity in Silico

To study the intrinsic heterogeneity[16] of a population of silicon model neurons, we designed a single chip that included 20 nominally identical isolated neurons using the MOSIS TSMC

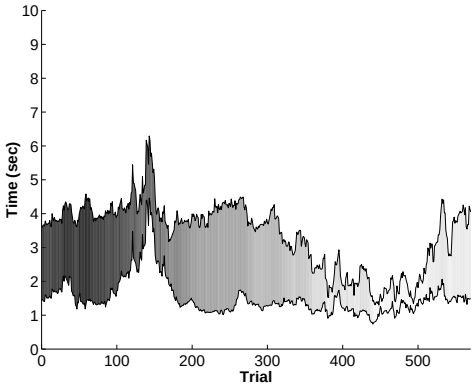
[16]This effect can also be referred to as built-in heterogeneity or natural offsets.



A.



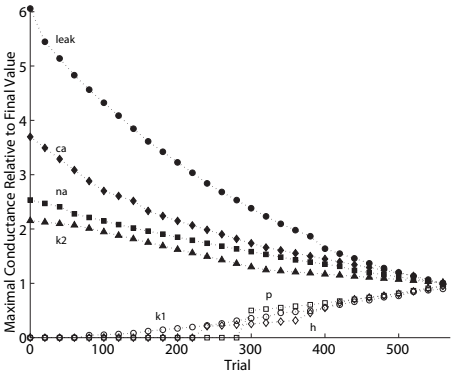
B.



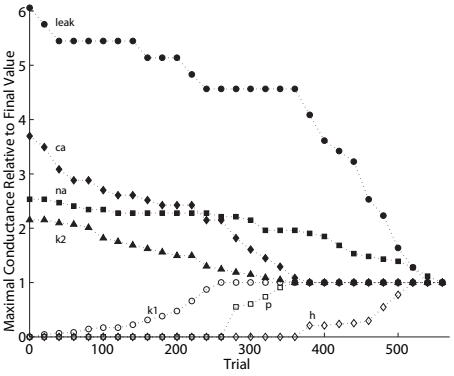
C.

Figure 53: Comparative analysis—progression of the silent time, spiking time, period, duty cycle and average spike frequency. The lower line represents silent time. The upper line represents the period. The difference between the upper and lower line represents the spiking time. The duty cycle is the ratio of the spiking time and the period. The color of the shading represents the average spike frequency (the darker the shading, the larger the average spike frequency). A. Uniform parameter variation method. B. Non-uniform parameter variation method. C. Random parameter variation method.

A.



B.



C.

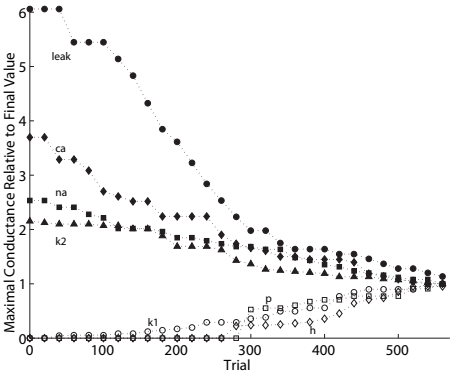


Figure 54: Comparative analysis—progression of maximal conductances relative to final values for every twenty trials. Note that the maximal conductances for $k1$, p, and h are shut off for Trial 0 (PBC model) and eventually turn on in a variety of ways. A. Uniform parameter variation method. B. Non-uniform parameter variation method. C. Random parameter variation method.

0.35 μm process (Section 2.2). The silicon model neurons were fabricated on a single silicon substrate, making them heterogenous by their physical nature. This intrinsic heterogeneity cannot be controlled[17] and is primarily derived from the following two sources of mismatch between supposedly identical transistors [43]: (1) variations in the location of the edges of the transistor caused by the limited resolution of the photolithographic process and (2) nonuniform conditions during the predeposition and diffusion of the impurities. For example, the differential-pair offsets in the E_{Leak} OTAs (Figure 55) will contribute to the variability in the minimum output voltages during bursting. From the figure, the voltage offsets from Neuron 10 and Neuron 19 are outliers, and the intrinsic heterogeneity implicit in that data will be manifested in the output data figures shown in Section 3.3.3. The framework of this section is as follows: In Section 3.3.1, we will describe our goal of mapping input and output spaces and how we obtained bursting outputs across the array of neurons. In Section 3.3.2, we will describe our *MidSearch* algorithm, which was integral for mapping the spaces. In Section 3.3.3, we will present the results from the four tests that we completed. In Section 3.3.4, we will conclude this section with descriptions of more tests that could have been performed and the shortfalls of this physical implementation.

3.3.1 Mapping Input and Output Spaces

For our application, we quantified the intrinsic heterogeneity in terms of the following measures: (1) *input* heterogeneity was derived from the distributions of the input voltage biases that controlled the model parameters (Table 3) and (2) *output* heterogeneity was derived from distributions of six time-based and voltage-based bursting output characteristics (Figure 56). Ideally, in the absence of intrinsic heterogeneity, the standard deviations of these distributions are zero. By definition, the silicon model neurons constitute a physical system, and consequently, the time-based output measures exhibited cycle-to-cycle variability. The input voltage biases, however, were fixed to millivolt precision, and the voltage-based output measures were relatively stable (*e.g.*, the relatively constant differential pair offsets were reflected in the minimum value of V_{mem}).

[17] Although this intrinsic heterogeneity can be decreased by layout techniques that improve transistor matching such as designing cross-coupled transistors. [47] [128]

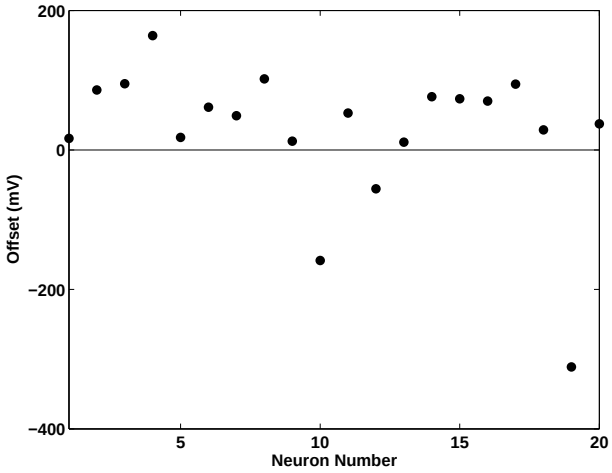


Figure 55: Intrinsic heterogeneity with the analog model—voltage offsets with the E_{Leak} differential pair.

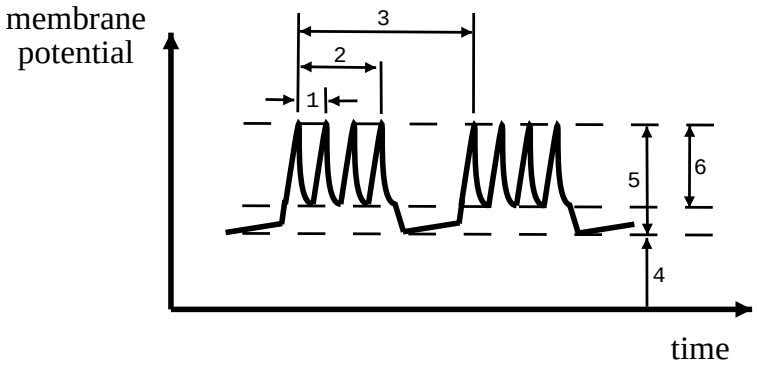


Figure 56: Intrinsic heterogeneity with the analog model—the output space parameters. (1) interspike interval, (2) spiking time, (3) period, (4) absolute minimum voltage, (5) total voltage amplitude, and (6) spike height. Note that the first three parameters are time-based and the last three parameters are voltage-based.

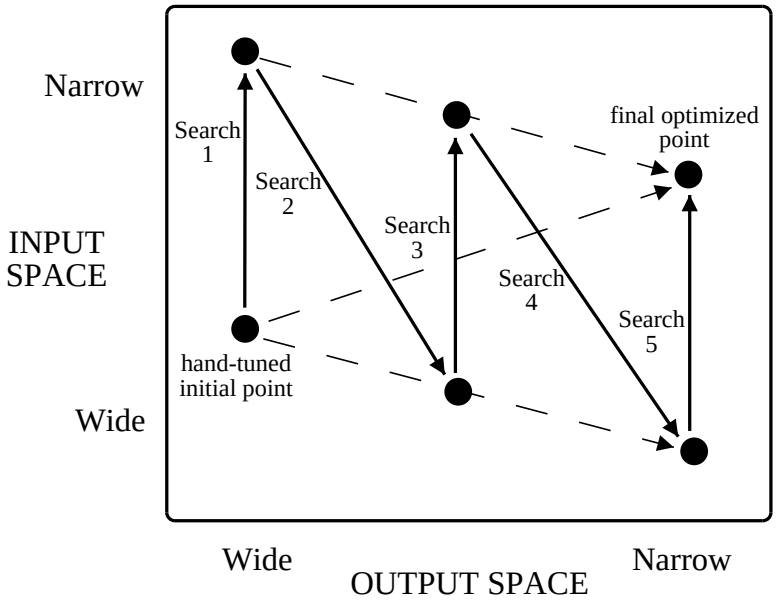


Figure 57: Intrinsic heterogeneity with the analog model—generalized approach for mapping the input and output spaces. Starting with the hand-tuned point, the final optimized point will be found by alternately narrowing the input and output spaces. The final optimized point will have narrower input and output spaces relative to the hand-tuned initial point.

As Figure 57 implies, the goal of our work was to map one space to another space,[18] and we expected the sizes of these spaces to be inversely related—as the size of the input space is constrained and thus decreases (increases), the size of the output space must increase (decrease) to maintain bursting across the entire neural array. With subsequent tests, our plan was to continue the process of successively reducing the input space and output space until we found the final optimized point as shown in the figure.[19] To reduce these input and output spaces, we employed a *MidSearch* algorithm that will be described in Section 3.3.2. This algorithm was also required to make fair comparisons between the input and output measures; otherwise, (arbitrary) sets of voltages would have been used to measure heterogeneity, resulting in much less meaningful results.

We first were required to manually obtain bursting outputs for each of the 20 neurons on our chip. As shown in Figure 5, a scanner was used to access the membrane potential,

[18] A proposal from the Laboratory for Neuroengineering (Bob Lee) submitted in October 2005 included a discussion of this general type of mapping problem that was to be addressed in the proposed work.

[19] This final optimized point is not an actual point; instead, it results in the smallest input space for an allowable output space.

V_{mem}, from a single selected neuron, but V_{mem} of Neuron 1 was also taken off chip to characterize the voltage-to-physiological relationships of each of the parameters. With these characterizations, we were able to determine a set of voltages that resulted in bursting for Neuron 1. Also note in Figure 5 that a single voltage controls each model parameter across all neurons, and in the absence of transistor mismatch (*i.e.*, the ideal case), this set of voltages that we found for Neuron 1 would also result in bursting for the other 19 neurons; because of transistor mismatch, this was not the case. We were able to use the set of voltages from Neuron 1, however, as a set of baseline input parameter voltages to provide us with a good general idea as to the set of voltages that should be used to obtain bursting with the other neurons. Consequently, we found 20 different sets of input voltages to obtain bursting in every neuron. These hand-tuned initial bursting points were not constrained, thus wide input and output spaces were generated. In general, considering the sensitivities of the input space measures, we expected the reversal potential (E) and half-maximal voltages (θ) to vary more from these baseline voltages than the maximal conductance ($\bar{g}$), sigmoidal slopes (σ), and time-constant voltages ($\bar{\tau}$).

To simplify the search for a bursting output, we first found subthreshold oscillations by using the eight subthreshold model parameters from the I_{Leak} and I_{NaP} currents[20] and turning off the spiking currents by setting their maximal conductances to zero (*i.e.*, $\bar{g}_{\text{Na}} = \bar{g}_{\text{K}} = 0$ nS). With an underlying subthreshold oscillation, we next varied the 12 voltages that controlled the spiking parameters to locate bursting for the remainder of the neurons. The resulting set of 20 voltages for Neuron n is given by the 20-parameter voltage array $(v_1^n, v_2^n, \ldots v_{20}^n)$ and this array was determined for all 20 neurons.

3.3.2 The MidSearch Algorithm

The following steps describe the *MidSearch* algorithm that we employed to implement the tests shown in Figure 57:

[20]The leak reversal potential, E_{Leak}, was fixed and provided the reference for all model parameter voltages. If all half-maximal voltages and reversal potentials were moved by the same voltage, then the membrane potential would not change. Therefore, one of these voltages can be (arbitrarily) defined as the reference and be set to a constant value. We chose E_{Leak} to be this voltage.

- To constrain the input (or output) space, set the allowable range for input (or output) parameter p to [minimum(v_p^a) maximum(v_p^b)] where, for this case, Neuron a required the minimum value v and Neuron b required the maximum value v. Re-calculate these allowable ranges after each pass through the algorithm.

- For a particular neuron, randomly select one of the parameters and sweep its voltage within its allowable range and determine the voltages for which it bursts (the criteria for bursting are discussed below). Move the voltage to the middle of the bursting range[21] and determine if the difference between the starting and ending voltages is within 15 mV (this threshold gives a good balance between testing time and a reasonable voltage change, including a consideration of electronic noise). If so, then do not select that parameter again until this test fails for another parameter.

- Repeat for the next randomly picked input parameter and continue until all 20 input parameters consecutively had voltage changes of less than 15 mV.

- See Figure 58A to see a cartoon of the method of moving voltages to the middle of a bursting range for two input parameters. Figure 58B shows how the midpoint is found for the maximal conductance ($\bar{g}$) and time constant ($\bar{\tau}$) parameters, which have an exponential voltage-to-physiological mapping; the reversal potential (E), half-maximal voltage (θ), and sigmoidal slope (σ) parameters are voltages and therefore did not require this type of mapping. Note that a $U_T/\kappa \approx 42$ mV voltage change represents an e-fold change in the physiological value (for $U_T = 25$ mV and $\kappa = 0.6$).

- Repeat for the next neuron and continue until all 20 neurons have completed. This concludes a single search as indicated in Figure 57.

- Repeat this algorithm until the ranges fail to decrease significantly. This is the final optimized point as indicated in Figure 57.

To determine the range of voltages in which bursting occurred, we started at a known bursting point and first increased the voltages to find the upper limit of the range and then

[21] We believed that the midpoint of this region made the neuron more robust to parameter variation.

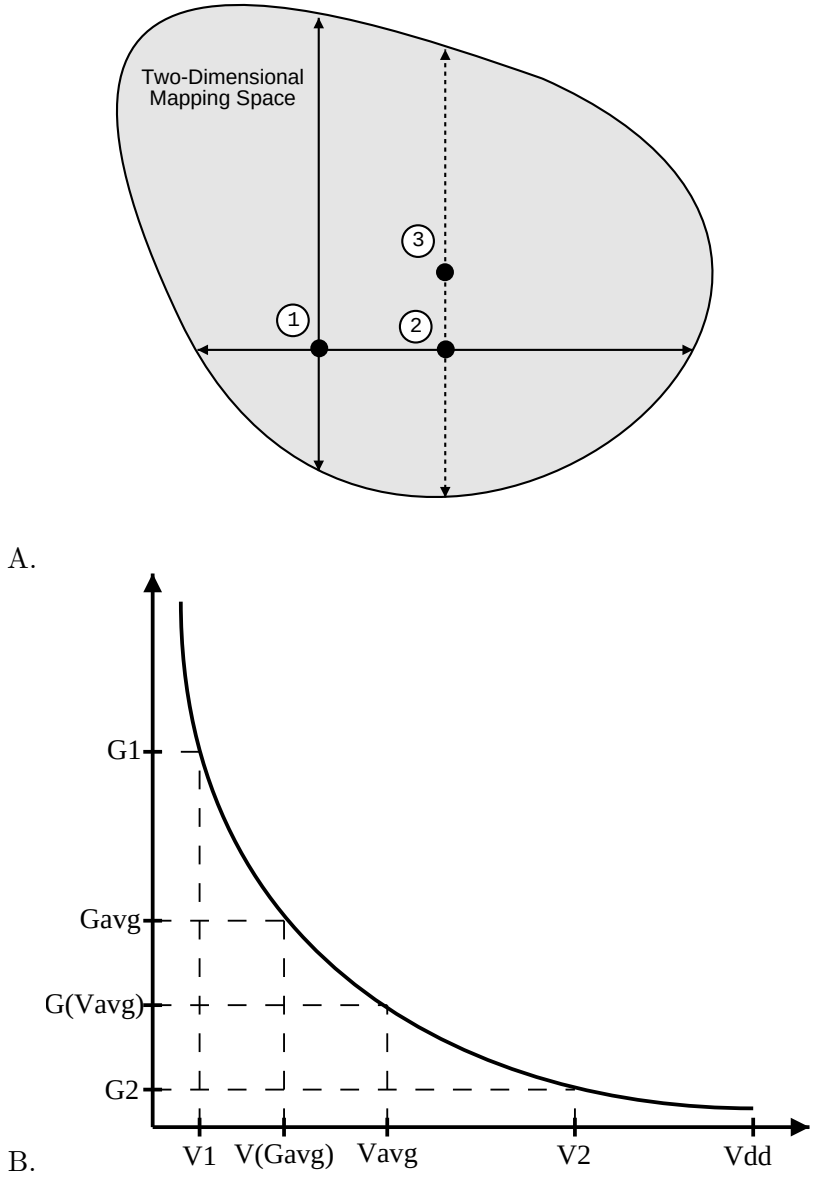


Figure 58: Intrinsic heterogeneity with the analog model—method used to adjust parameter voltages. A. The two-dimensional mapping space is used for simplicity. Point 1 gives the original points. Point 2 gives the intermediate point after sweeping one parameter and finding the midpoint of its bursting range. Point 3 gives the final point after sweeping the other parameter and finding the midpoint of its bursting range. B. $V1$ and $V2$ represent the ends of the bursting range, and $G1$ and $G2$ correspond to their physiological values, respectively. The midpoints of these parameters are first found in the physiological space and then these values are converted back to voltages (similarly for the time constant parameters). Therefore, the midpoint voltage is different from the average of $V1$ and $V2$.

decreased the voltages to find the lower limit of the range. At each voltage, we recorded 10 seconds of data, and averages were calculated for the following bursting measures: (1) starting spike frequency, (2) ending spike frequency, (3) spike frequency, (4) spikes per burst, (5) spiking time, (6) silent time, (7) duty cycle, (8) standard deviation of the peak voltages, and (9) amplitude. Relative and absolute thresholds were used to classify the data as bursting. For the relative thresholds, the differences between the last good bursting data and the current data had to be less than 30 percent for measures $(1) - (7)$. In addition, absolute thresholds had to be achieved for measures $(6) - (9)$ to avoid a misclassification with tonic firing and "degenerate" bursting. These bursting-measure thresholds allowed for reasonable incremental changes in the bursting characteristics.

Because we were working with a physical system (*i.e.*, a silicon neuron), we had to consider noise when measuring the output space. For example, we could not expect to obtain a duty cycle of exactly 50.00000... percent. Instead, a reduced, or constrained, value could be defined as $48-52$ percent. This noise is introduced not only by the physical system but also by the sampling-time tradeoff—the shorter the time, the greater the variability (noise) in the bursting characteristics data (the output space), but the more data that can be obtained.

As a result of the *MidSearch* algorithm, each final set of input parameter voltages for each neuron was found using a single, well-defined algorithm.[22] Because of this voltage-normalization process, we were also able to make fair comparisons and thus quantify the intrinsic heterogeneity between the neurons using non-arbitrary bursting points in parameter/voltage space.

Finally, during our preliminary testing, we determined that I_{K} was largely irrelevant for some bursting points; that is, either $\bar{g}_{\mathrm{K}}$ was too small or $\theta_n(I_{\mathrm{K}})$ was too large or both. In these instances, comparisons could not be made across neuronal input parameters or output measures. As a result, to make I_{K} relevant, we implemented the following constraints before

[22] We would have preferred to write that a bursting point is located at the center of a bursting region, but we were unable to prove this because we were working with a 20-dimensional space. In addition, we also believed that our algorithm increased the aggregate bursting ranges and resulted in a more stable bursting point, but we could not prove this either. Finally, our algorithm included a process to randomly select parameters, and therefore, a deterministic final point was not found.

we began collecting data from the tests: (1) $\bar{g}_{\mathrm{K}} < 2.75$ V and (2) $1.52 < \theta_n(I_{\mathrm{K}}) < 1.81$ V.

3.3.3 Results

After four tests, we obtained four sets of parameter voltages that resulted in bursting for every neuron, and these tests will be referred to as the following in this document:

- Test 0: original bursting points from the manually hand-tuned set of voltages
- Test 1: after first pass of *MidSearch* algorithm (corresponds to Search 1 in Figure 57)
- Test 2: after second pass of *MidSearch* algorithm (Search 1 repeated)
- Test 3: after third pass of *MidSearch* algorithm (Search 2)

We recorded data from each neuron using the set of voltages determined at the end of each of the four tests to determine the bursting characteristics (Figure 59, Figure 60, Figure 61, and Figure 62). Each plot shows 2.5 seconds of data, and this time frame was selected to capture at least one full bursting cycle for each neuron. The membrane potential, V_{mem}, is the y-axis data. Every plot uses the same 350-mV range, which corresponds to a chip voltage of $1.55 - 1.90$ V. This voltage range was selected to best capture the full output ranges of all of the neurons for all of the tests. Note that these plots show a wide variety of bursting characteristics such as period, duty cycle, and spike heights. Also, as expected from Figure 55, the minimum voltages from the outputs from Neuron 10 and Neuron 19 are noticeably smaller than those from the other neurons because of the differential-pair offsets in the E_{Leak} OTAs; similarly, the minimum voltage from the output from Neuron 4 is larger.

The voltage ranges of the input parameters for the four tests are shown in Figure 63, and Table 16 shows related information in tabular form. Relatively wide input parameter voltage spaces were generated by Test 0 (the unconstrained hand-tuned initial bursting points) in which the smallest range for any parameter was 58 mV. After Test 2 (the second test to reduce the input space), 14 of the 20 parameters had voltage ranges ≤ 51 mV. The only parameter types that required larger ranges were the reversal potentials and the half-maximal voltages. In fact, six of the input parameters had voltage ranges ≤ 5 mV (E_{Na}, $\bar{g}_{\mathrm{K}}$,

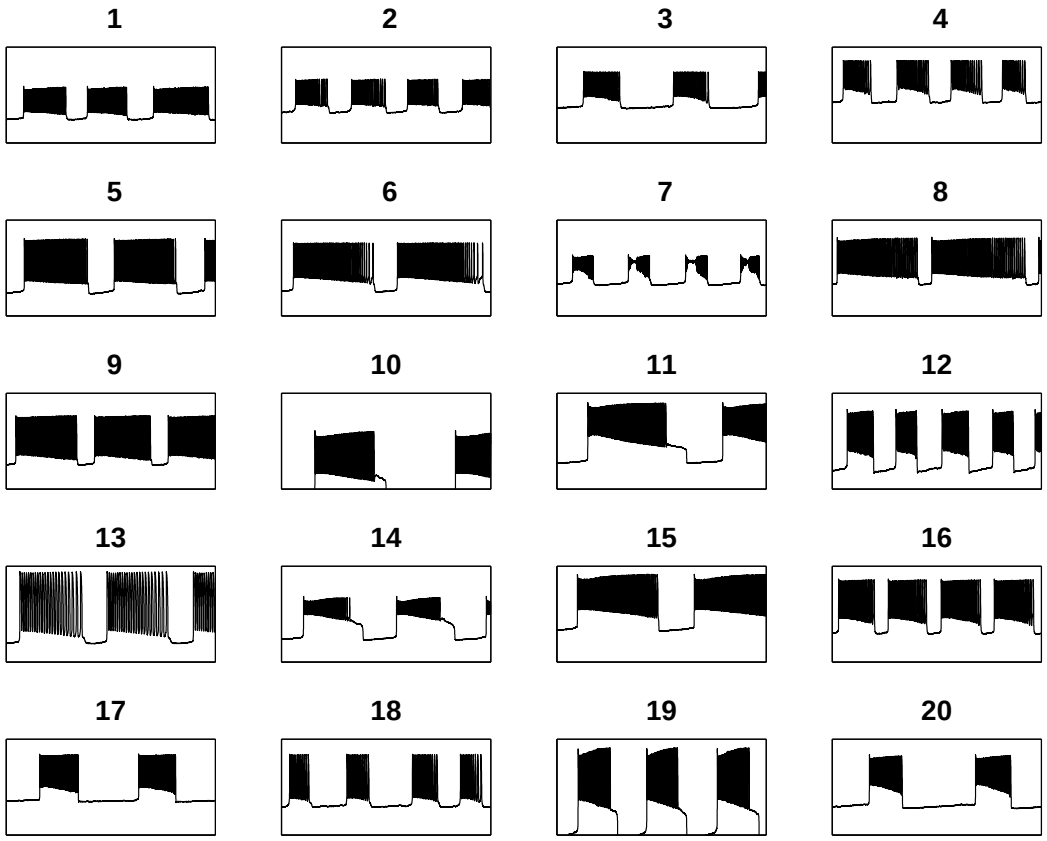


Figure 59: Intrinsic heterogeneity with the analog model—neural outputs from Test 0.

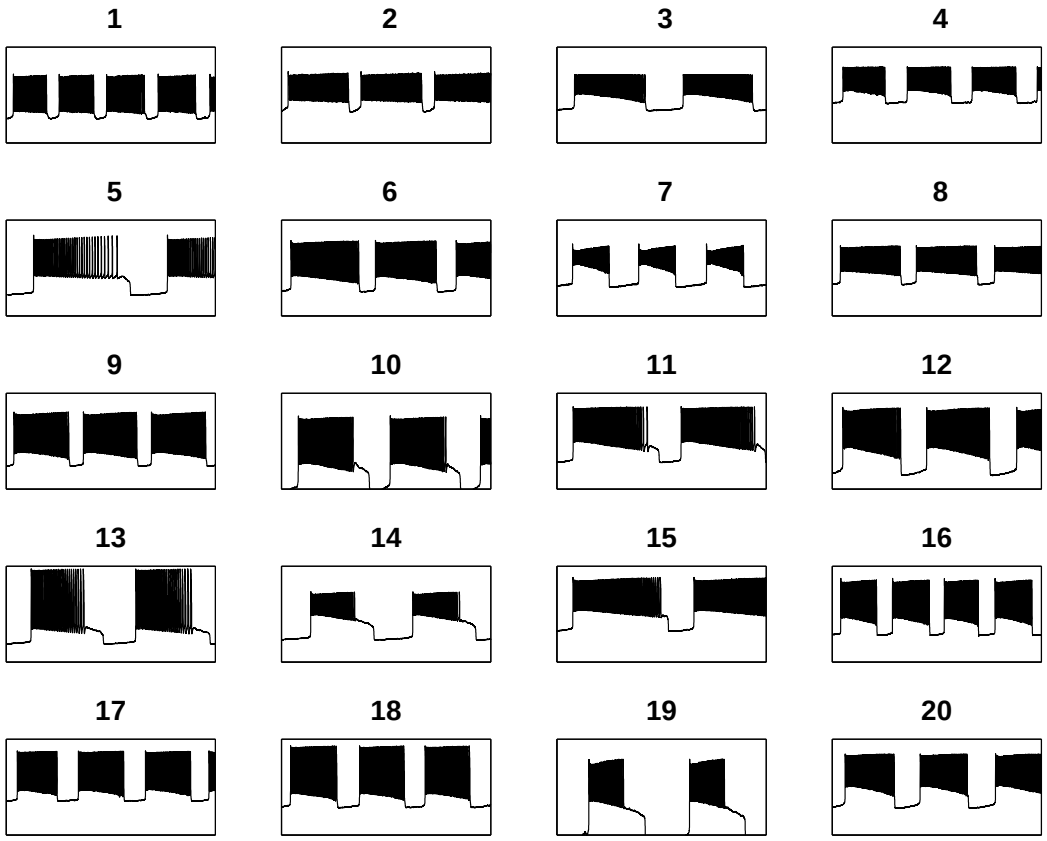


Figure 60: Intrinsic heterogeneity with the analog model—neural outputs from Test 1.

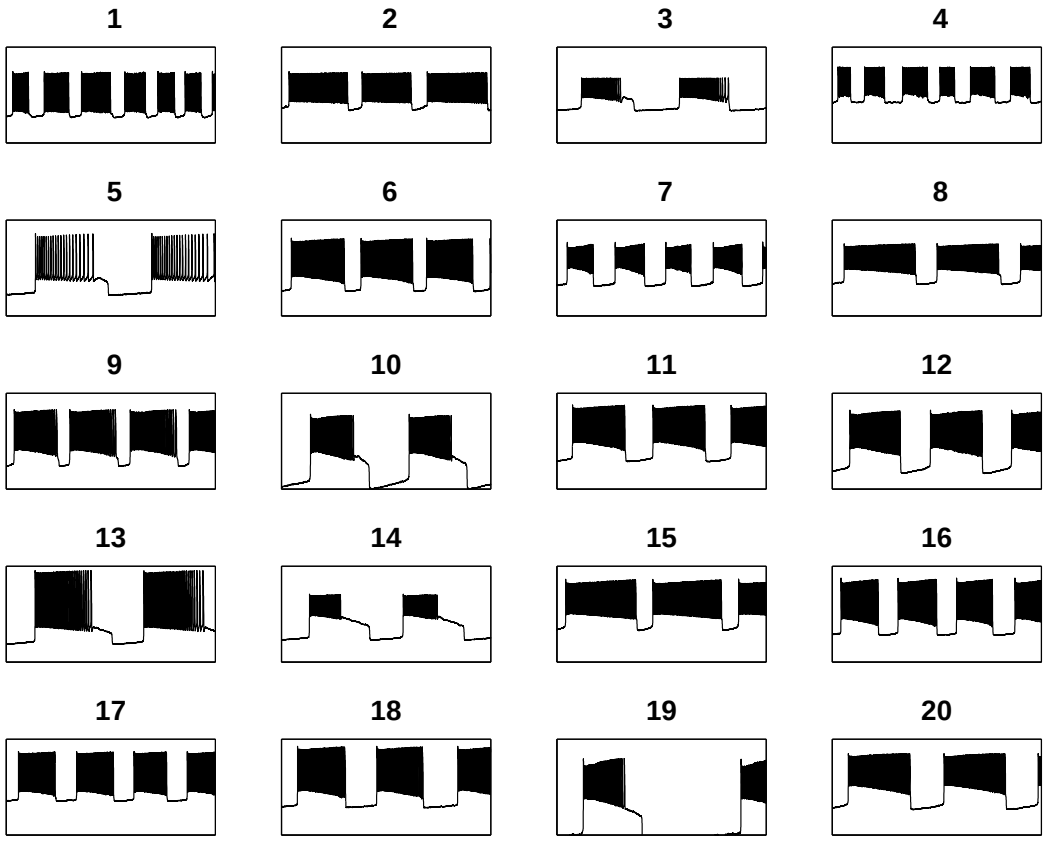


Figure 61: Intrinsic heterogeneity with the analog model—neural outputs from Test 2.

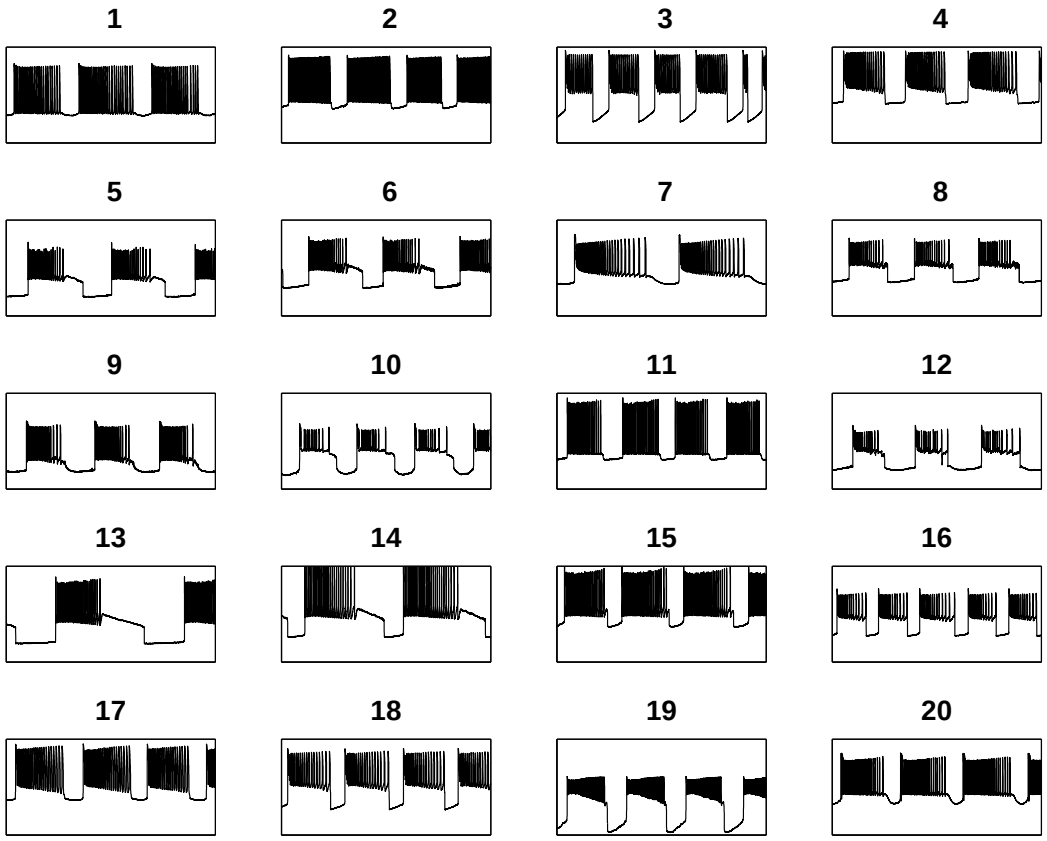


Figure 62: Intrinsic heterogeneity with the analog model—neural outputs from Test 3.

σ_h, $\sigma_n(I_{\mathrm{Na}})$, $\sigma_n(I_{\mathrm{K}})$). By design, as shown in Table 16, the ranges and standard deviations of each of the 20 input parameters reduced after Test 1 and again after Test 2. As expected, these input parameter statistical measures increased after Test 3, which was designed to reduce the output space. Note that, in general, the reversal potential (E) and half-maximal voltages (θ) have larger ranges to maintain bursting than the maximal conductance ($\bar{g}$), sigmoidal slopes (σ), and time-constant voltages ($\bar{\tau}$).

The ranges of the output-space parameters (Figure 56) for the four tests are shown in Figure 64, and Table 16 shows related information in tabular form. Note that these output parameters are not deterministic with our physical system because of cycle-to-cycle variability—statistics taken on new sets of output data give results within ten percent of those shown in the table. Note that for Test 1 and Test 2, as the input space was reduced, the output space also tended to reduce; for Test 3, however, as the output space was reduced, the input space increased significantly. Figure 65 is an updated version of Figure 57 that reflects the tests that we actually completed and the results that we obtained.

Note that we did not address the problem of non-uniqueness; that is, the final set of input and output parameters satisfies our constraints, but it is certainly a non-unique solution in the 20-parameter input space. Because we are working with an under-determined problem and an algorithm that includes the random selection of parameters, we could increase the number of constraints to reduce the number of possible sets of voltages in solution space, but we could not fully eliminate the non-uniqueness issue—a deterministic final point is unlikely to be found. In addition, a final point is not guaranteed to be the "best" point in terms of a stable bursting region; for example, a final point could be the result of irreversibly falling into a region of parameter space because of the constraints placed on the algorithm.

3.3.4 Additional Testing

Because of time constraints, we did not complete as many passes of the *MidSearch* algorithm as we planned. Given more time, we could have used the set of input parameter voltages and output bursting measures after each pass of the *MidSearch* algorithm to analyze the following theories:

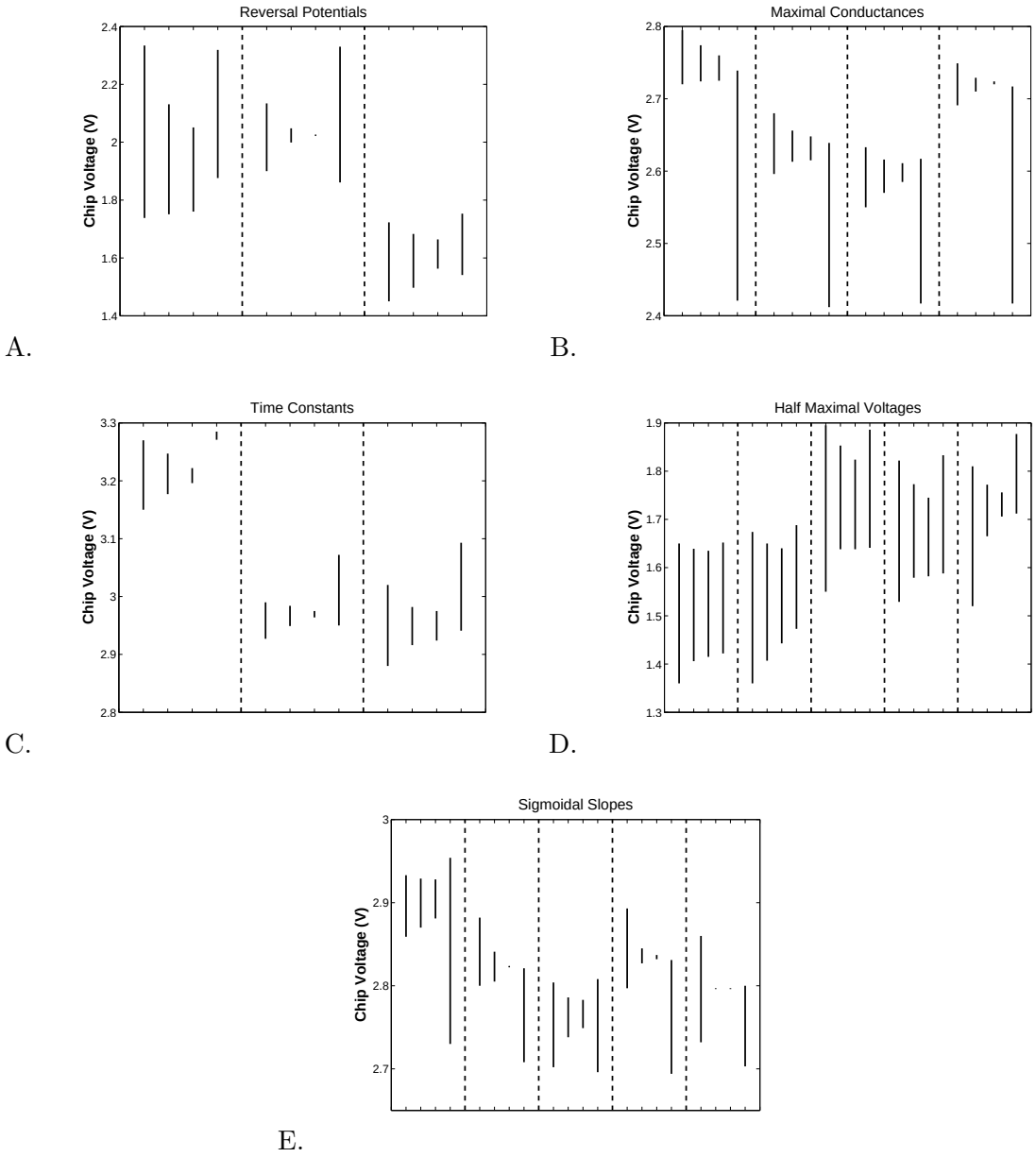


Figure 63: Intrinsic heterogeneity with the analog model—input parameter voltage ranges for all neurons after each test. Parameter order of each subfigure: A. Reversal Potentials ($E_{\mathrm{Na}}(I_{\mathrm{NaP}})$, E_{Na}, E_{K}), B. Maximal Conductances ($\bar{g}_{\mathrm{Leak}}$, $\bar{g}_{\mathrm{NaP}}$, $\bar{g}_{\mathrm{Na}}$, $\bar{g}_{\mathrm{K}}$), C. Time Constants ($\bar{\tau}_h$, $\bar{\tau}_n(I_{\mathrm{Na}})$, $\bar{\tau}_n(I_{\mathrm{K}})$), D. Half-Maximal Voltages (θ_m, θ_h, θ_q, $\theta_n(I_{\mathrm{Na}})$, $\theta_n(I_{\mathrm{K}})$), and E. Sigmoidal Slopes (σ_m, σ_h, σ_q, $\sigma_n(I_{\mathrm{Na}})$, $\sigma_n(I_{\mathrm{K}})$). The maximal conductance and time-constant parameters have logarithmic voltage-to-physiological relationships. The other parameters have linear relationships.

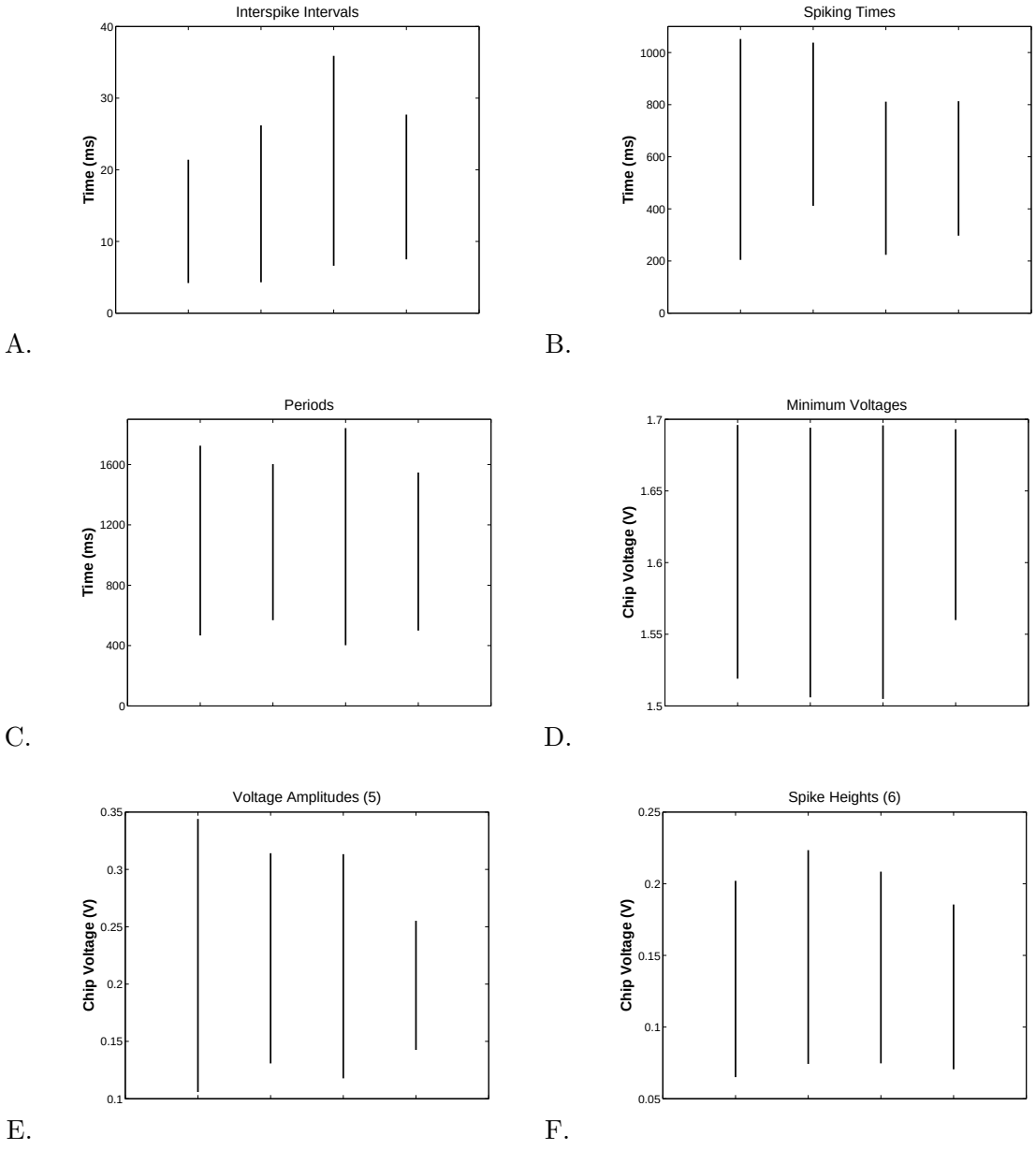


Figure 64: Intrinsic heterogeneity with the analog model—output parameter ranges for all neurons after each test. The number in parenthesis in each of the subfigure titles corresponds to the output space parameter shown in Figure 56.

Table 16: Intrinsic heterogeneity with the analog model—ranges and standard deviations of each input and output parameter after each test. The units of the input-space parameters and the last three output-space parameters are mV. The units of the first three output-space parameters are ms.

Parameter	Test 0		Test 1		Test 2		Test 3	
	Range	STD	Range	STD	Range	STD	Range	STD
Input-Space Parameters								
$E_{\mathrm{Na}}(I_{\mathrm{NaP}})$	596	149	380	78	291	64	443	105
E_{Na}	234	61	49	16	0	0	469	110
E_{K}	273	73	186	51	101	33	212	60
$\bar{g}_{\mathrm{Leak}}$	75	22	50	16	35	11	318	106
$\bar{g}_{\mathrm{NaP}}$	84	26	43	10	33	6	227	83
$\bar{g}_{\mathrm{Na}}$	83	26	46	12	26	6	200	77
$\bar{g}_{\mathrm{K}}$	58	17	19	5	4	1	300	115
$\bar{\tau}_h$	120	29	70	13	26	5	14	3
$\bar{\tau}_n(I_{\mathrm{Na}})$	63	15	35	9	11	2	122	45
$\bar{\tau}_n(I_{\mathrm{K}})$	140	43	66	14	51	11	152	55
θ_m	290	72	233	67	220	63	230	72
θ_h	314	83	243	71	197	59	215	61
θ_q	345	84	215	61	186	49	245	66
$\theta_n(I_{\mathrm{Na}})$	293	79	194	64	163	48	245	71
$\theta_n(I_{\mathrm{K}})$	290	76	107	29	50	13	165	45
σ_m	74	22	59	16	47	12	224	62
σ_h	82	24	36	9	0	0	113	32
σ_q	102	27	48	13	34	7	112	33
$\sigma_n(I_{\mathrm{Na}})$	96	27	18	7	5	1	137	37
$\sigma_n(I_{\mathrm{K}})$	128	35	1	0	1	0	97	28
Output-Space Parameters								
interspike interval	17	5	22	5	29	6	20	5
spiking time	847	278	626	189	588	167	516	120
period	1258	359	1034	280	1439	341	1048	253
minimum voltage	176	44	188	44	191	43	133	32
voltage amplitude	238	55	183	50	196	49	113	30
spike height	137	34	149	38	134	33	115	35

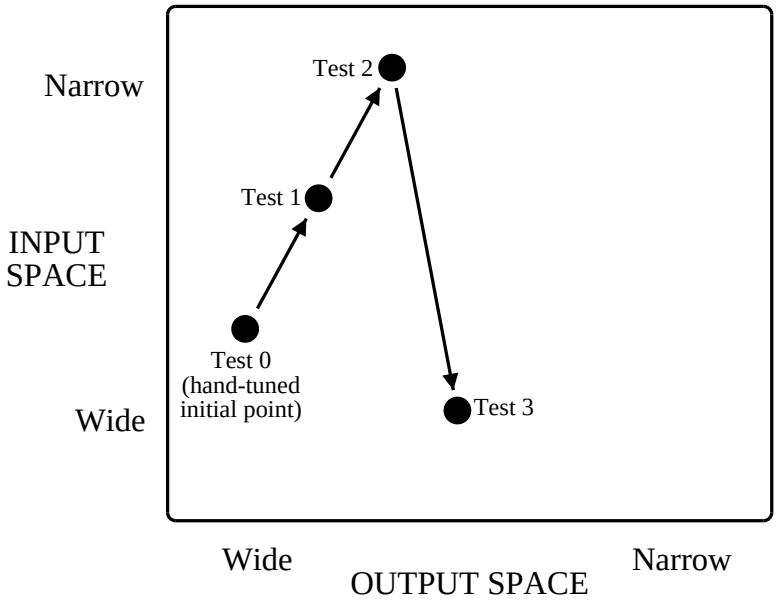


Figure 65: Intrinsic heterogeneity with the analog model—realized mappings of the input and output spaces.

Tests of Mean Parameter Values. By setting each parameter to its mean value across the array, we could determine the number of neurons that are bursting, silent, or tonically firing and determine how this number changed with each pass through the algorithm. We could also quantify the distances that each parameter is from these mean values to determine the distance that each bursting point is from the center of the 20-parameter space. We have to be careful with these distance measures because we would be required to combine terms with a variety of units (V, s, nS); a simple summation of maximal conductances could even be problematic if one of them influences the result in a disproportionate way. Finally, we could also determine if a relationship exists between the output burst characteristics and the distance measures.

Tests for Subthreshold Oscillations. By turning off the spiking maximal conductances (*i.e.*, $\bar{g}_{\mathrm{Na}} = \bar{g}_{\mathrm{K}} = 0$ nS), we could determine how many neurons have an underlying subthreshold oscillation and determine if this result correlates with other results such as larger bursting parameter ranges.

Tests for Bursting Regions. By completing a few burst mapping diagrams (*e.g.*, determine the bursting region in $\bar{g}_{\mathrm{NaP}} - E_{\mathrm{Leak}}$ space), we could determine the different sizes across

the neural array to provide us with another bursting output measure. We could determine if this result correlates to the results from the tests above or to the bursting characteristics. For example, we could determine if the duty cycle has a negative correlation to the size of the burst mapping diagram. In addition, we could determine how many diagrams overlap, but we have to be careful with the interpretation because we would only be considering a two-dimensional slice of a 20-dimensional space. We could take this additional measure of intrinsic heterogeneity in the output space and determine if it is too large compared to the applied heterogeneity that is generated with the heterogeneity circuit (Section 2.2.4). Because the bursting space has 20 dimensions, determining whether it is convex, hyperellipsoid, or some other shape is an intractable problem; as a result, we are forced to do analysis using two-dimensional burst mapping diagrams. Alternatively, in terms of a reasonable "return on investment," we could perform a random sampling of the input parameter space to provide a characterization of the output space and perhaps obtain contours of bursting regions, including regions with local maxima; this result would probably be acceptable for the aVLSI community, but not for the biological science and modeling communities.

3.4 Significance and Future Directions

We tested single-neuron models using both digital (computational) and silicon (physical) implementations. We described and tested a straightforward and efficient search algorithm that achieved our goal of reliably locating a desired neural behavior in the vast multi-parameter space of a conductance-based HH neuron model. The desired behavior was bursting, and to classify this type of neural dynamics, we used a simple cost function whose inputs were derived from the frequency content of the output. This cost function was used with a stochastic gradient descent-type algorithm to locate parameter values that allowed the neural model to produce bursting within a specified tolerance. We demonstrated good results, including those showing that the utility of our algorithm improved as the pre-defined allowable parameter ranges increased. For example, using the initial parameter ranges, our algorithm did not perform better than a random search, but as the parameter ranges were extended, our method proved to be computationally efficient. This result was

expected because in an extreme case (*i.e.*, for a case in which the pre-defined parameter ranges were virtually zero), our algorithm would not aid a search for bursting points, but for some set of pre-defined parameter ranges, our search algorithm would perform better than a random search. For our model, this threshold existed between the 100 percent and 120 percent parameter ranges. By finding hundreds of points in parameter space that resulted in bursting that had frequency components similar to those found in the output of the model at the canonical point,[23] we were able to show that the "canonical" point was really arbitrary from a modeling perspective. This result was important and was later tied to FPGA results that showed that the size of the bursting region could be increased by moving to a different point in parameter space. In addition, our algorithm was also used to demonstrate that the bursting region is a contiguous region of parameter space, and this result led us to the silicon model neuron tests that related two dynamically different bursting models through an exploration of parameter space. Our algorithmic approach to this general problem was based on the understanding that real-world problems need real-world solutions fast and approximately, but not exactly, correct. To search a vast parameter space, a brute-force method is not feasible, every possibility does not have to be tested, and cases can be discarded without checking them individually. In addition, solutions that lie near one another are similar in some respect, and the cost function provides the "quality" of the solution.

Using the parameter-search techniques that we previously developed, we used the silicon model neuron to demonstrate two important findings. We first validated the flexibility of a previously published silicon-neuron architecture [106]. We accomplished this goal by implementing the various functional behaviors of the HN, PBC, and AC conductance-based neural models with this single integrated-circuit chip. One of the criticisms of neuromorphic engineering is that the circuits that are designed to emulate biology are too specific. Our findings, however, illustrated the value of this silicon-neuron architecture as a general neural-modeling tool and showed that opportunities exist to demonstrate the applicability of these

[23] Arguably, an infinite number of points exist if not for the finite precisions of the parameters of the digital model.

neuromorphic architectures to a broad set of problems. Although we only demonstrated our results using an architecture designed for just a single neuron, we will be able to study neural populations and the effects of varying like parameters in the same amount of testing time to obtain other scientific results; this scalability is an important aspect of working with a physical system. We also used the silicon model neuron to relate two dynamically different bursting models through an exploration of parameter space. These bursting models, PBC and HN, are different because their bursting mechanisms rely on different ion channels—the PBC model uses a persistent Na^+ channel and the HN model uses a Ca^{++} channel and an h channel. We determined that these models represent points in a contiguous bursting space that spans between the two models—from one set of parameters to another set of parameters. Although we did not have a good reason to presume this specific interesting result, it does support our broad hypothesis that bursting regions are contiguous and well behaved. Interestingly, we also found that the point in parameter space that represents the average parameter values of these two canonical bursting points is also a bursting point, although this is certainly not a general result.

To continue our understanding of heterogeneity at a population level, we fabricated an array of 20 isolated silicon model neurons. We demonstrated that a generalized algorithmic approach for mapping the input and output spaces of the individual neurons could be accomplished. To do so, we relied on the property that burst-mapping regions are contiguous, which we demonstrated in earlier studies. One of the objectives of our work was to locate the "center" of a multi-dimensional bursting region because we believed that this point would result in increases to the aggregate bursting ranges of the parameters, resulting in a more stable bursting point; this theory, however, is unproven largely because of the size of the parameter space. We know that each neuron etched in silicon on the same substrate will be different because of the inherent offsets in their physical structures. Although offsets from transistor mismatch may be useful in some applications [66], we did not benefit from this result. As a result, a manifestation of the intrinsic heterogeneity with the silicon model neuron is parameter variability, and this variability proved to be too large to satisfy our larger goal of studying the role of "controllable" heterogeneity in rhythmic networks

of neurons using this physical implementation. Therefore, we decided to use a different technology platform to pursue this work. Nonetheless, we were able to demonstrate that the input and output spaces of a silicon model neuron could be optimized, but that the parameter variability was too large to construct networks of these neurons.

CHAPTER IV

ANALYSIS OF HETEROGENEOUS NETWORKS

Real neurons, as opposed to deterministic model neurons, can exhibit a variety of behaviors as a result of their nonidentical intrinsic membrane properties [109], and in fact, these membrane parameters, such as maximal conductances, possess considerable variability [9]. We wanted to test these beliefs by using the FPGA model neuron to better understand how rhythmicity in an HCO is generated, maintained, and eliminated by the introduction of parametric and topological heterogeneity—that is, we wanted to determine whether heterogeneity is functionally advantageous. From the literature, heterogeneity has been shown to have both positive and negative consequences on network rhythmicity, which is essential to locomotion and other complex oscillatory neural behaviors (Section 1.1.1). We expected to find regions of parameter space in which individual neurons would not burst endogenously, but would exhibit anti-phasic bursting when connected in an HCO configuration. In addition, we hypothesized that the heterogeneity imposed on this rhythmic network of neurons would result in positive changes in measures of output robustness—specifically to the sizes of a variety of two-dimensional burst-mapping diagrams.

Regarding network rhythmicity, anti-phasic bursting was the most relevant network behavior for our study because of our interest in rhythmic pattern-generating systems, and we expected that a study of the anti-phasic bursting dynamics would show that heterogeneity leads to network robustness because of population effects. This study also continued our analysis of bursting regions (Chapter 3) in which the dynamical output of interest was stable bursting, but also incorporated the role of coupling to investigate synchronization in a population of neurons with heterogeneous properties—that is, we studied the simultaneous effects of coupling and parameter heterogeneity on the dynamics of the network. Therefore, we were always required to determine the number of neurons in the network that were bursting to determine the extent that network rhythmicity existed in a particular

region of parameter space. In addition, we were interested in studying the effects of ipsilateral excitatory coupling and the role of sparsely, moderately, and fully connected reciprocal (contralateral) inhibition.

We employed the FPGA model neuron for this testing because of its flexibility, accuracy, and speed. The flexibility derived largely from the control of the N^2 synaptic parameters and the ability to tune heterogeneity to a desired level. In addition, this real-time, fixed-point model gave single-neuron results that matched closely to those of a floating-point SIMULINK model (Section 2.3). We began our analysis using a homogeneous configuration to provide us with control HCO data that was used for comparisons with the data from a variety of heterogeneous configurations. We referred to homogeneity as the ideal, symmetrical, or zero-heterogeneity case, in which all neurons in the fully connected network were identical and interchangeable, and an absence of these conditions implied the presence of heterogeneity. The data of interest (the system output) was the size of the burst-mapping diagram for three two-dimensional intrinsic spaces because this provided a measure of synchronization in different regions of parameter space. We chose multiple diagrams because we did not want to bias our results to one region of parameter space, and we constructed these diagrams from the three intrinsic parameters that we thought were the most critical for bursting ($\bar{g}_{\mathrm{NaP}}$, $\bar{g}_{\mathrm{Leak}}$, E_{Leak}). We believed that changes to the sizes of these diagrams provided us with good indications of the robustness of the network in response to a variety of heterogeneity tests in which the heterogeneity was specific and quantifiable (the system input). We expected the burst-mapping diagrams of single neurons in the homogeneous configuration to be identical, but by introducing intrinsic parametric heterogeneity, we expected the individual bursting regions to shift in parameter space because of the resulting heterogeneity in the firing properties of the individual neurons. We expected these offsets to the burst-mapping diagrams to be advantageous in a connected network because they resulted in an effective bursting region that covered a larger part of parameter space than in the ideal (homogeneous) case, making the network more robust to parameter variation.

4.1 Experimental Methods

To setup the analysis that we used to study heterogeneity using the fully connected HCO configuration that was implemented with the FPGA model neuron, we must first describe our experimental methods, including such topics as configurations, definitions, and justifications. We will describe the various network configurations (homogeneous versus heterogeneous), including the parameters that were varied, and then follow with descriptions of other terms relevant to our testing such as heterogeneity, bursting, coherence, robustness, and connectivity. Included in this discussion is a description of how we obtained the normal distributions required to impose parametric heterogeneity in the network and the justifications that we made to use specific two-dimensional burst-mapping diagrams. In addition, we will describe the tests that we completed and the format of the results that we used.

Configurations. We defined a *homogeneous* configuration as one in which the all intrinsic and synaptic model parameters were identical, but with one exception—all ipsilateral excitatory connections were implemented with the same synaptic strength and reversal potential (0 mV) and all contralateral inhibitory connections were implemented with the same synaptic strength and reversal potential (−80 mV). The canonical set of model parameters were defined as those that were used in the published equations (*e.g.*, $\bar{g}_{\mathrm{NaP}} = 2.8$ nS, $\bar{g}_{\mathrm{Leak}} = 2.8$ nS, and $E_{\mathrm{Leak}} = -60$ mV) [8]. We considered this as the ideal case—ideal in the sense of an absence of the built-in heterogeneity that we observed with the silicon model neuron (Section 3.3). We defined a *heterogeneous* configuration as one that did not have identical intrinsic or synaptic parameters across the network, such as ones in which a normal distribution was applied to an intrinsic parameter or in which the synaptic connections did not form a fully connected symmetrical topology ($\bar{g}_{\mathrm{syn}-i,j} \neq \bar{g}_{\mathrm{syn}-j,i}$). For all of our testing, we only varied the following five parameters, which consisted of three intrinsic and two synaptic parameters (four maximal conductances and one reversal potential): $\bar{g}_{\mathrm{NaP}}$, $\bar{g}_{\mathrm{Leak}}$, E_{Leak}, $\bar{g}_{\mathrm{syn-i}}$, and $\bar{g}_{\mathrm{syn-e}}$.

The differences between the homogeneous and heterogeneous HCO configurations are summarized in Table 17. These forms of heterogeneity can be separated by two types of variation—parametric (first two rows) and topological (last row). Parametric heterogeneity

Table 17: FPGA HCO configurations—comparison between homogeneous and heterogeneous definitions.

	Homogeneous	Heterogeneous
Synaptic Parameters	identical $\bar{g}_{\mathrm{syn-i}}$, $\bar{g}_{\mathrm{syn-e}}$	distributed $\bar{g}_{\mathrm{syn-i}}$
Intrinsic Parameters	identical	distributed
Topology	fully connected (symmetrical)	not fully connected (asymmetrical)

includes synaptic (external) and intrinsic (internal) forms. Note that an external input, I_{app}, could have also been considered as a synaptic input—that is, external to the neuron (*e.g.*, a sensory input). The homogeneous configuration makes all-to-all synaptic connections; that is, it is a fully connected, symmetrical network. Topological variation occurs when some of the weights of the synaptic connections are set to zero, resulting in an asymmetrical network. We quantified all of these forms of heterogeneity for every test that we conducted.

Heterogeneity. We defined *heterogeneity* as the coefficient of variation of a distribution of intrinsic parameter values—that is, the standard deviation normalized by the mean. For example, if we applied x% heterogeneity to a maximal conductance, $\bar{g}$, using a targeted distribution with a mean of $\mu_{\mathrm{g-t}}$ and a standard deviation of $\sigma_{\mathrm{g-t}}$, then the resulting statistical measures from the random normal distribution are required to give the following mean and coefficient of variation:

$$\mu_{\mathrm{g-a}} = \mu_{\mathrm{g-t}} \pm 2\% \tag{64}$$

$$c_{v,\mathrm{g-a}} = \frac{\sigma_{\mathrm{g-a}}}{\mu_{\mathrm{g-a}}} \times 100\% = x \pm 2\% \tag{65}$$

where the $_{\mathrm{-t}}$ and $_{\mathrm{-a}}$ represent the targeted and actual quantities, respectively. To be as consistent as possible with this heterogeneity measure, we were required to make an adjustment to the distribution of a voltage parameter, specifically E_{Leak}. Because the bursting ranges of the maximal conductances that we used had coefficient of variations of approximately 40 percent, we chose a coefficient of variation for E_{Leak} that corresponded to its bursting range (3 percent) for these "40-percent heterogeneity" tests. We scaled appropriately for larger or smaller heterogeneity. For each grid point in a burst-mapping diagram, we obtained a new random distribution to *average* the heterogeneity results, giving unbiased

results that would have otherwise not occurred from using the same normal distribution for every grid point. We believe that our method of quantifying heterogeneity is an important distinction in our work because the term heterogeneity is not well defined in the literature (Section 1.1.1).

The normal distributions that we required for the parametric heterogeneity testing were randomly distributed around intrinsic parameter values. We validated the normal, or Gaussian, distribution using the *rankit* method. For example, to apply heterogeneity to every $\bar{g}_{\text{Leak}}$ in a 36-neuron HCO, we first obtained 36 values from a standard normal distribution ($\mu = 0$, $\sigma = 1$) using the MATLAB command `randn` and ordered them from smallest to largest. Next, we obtained 36 values from a non-standard normal distribution with a target mean and standard deviation and ordered them from smallest to largest; because maximal conductances must be positive quantities, we changed all negative values from the distribution to zero. We then fit a line through the two data sets and measured the R-squared value. The non-standard normal distribution was chosen for the heterogeneity test for a single grid point if it satisfied the following three criteria: (1) The mean was within two percent of the target value (Eq. (64)). (2) The standard deviation was within two percent of the target value (Eq. (65)). (3) The R-squared value was greater than 98 percent. In addition, because we used nine-bit precision ($2^{-9} = 0.002$) with the maximal conductance parameters, we rounded each value in the data set to the nearest 0.001 nS.

Synaptic-Weight Model. We normalized the inhibitory and excitatory synaptic weights, $\bar{g}_{\text{syn}-\text{i}}$ and $\bar{g}_{\text{syn}-\text{e}}$, respectively, to decouple size effects so that accurate comparisons could be made between the results from tests in which topological heterogeneity was imposed. We referred to this normalization process as the *synaptic-weight model*, and it is shown pictorially in Figure 66. The most important criteria is that the area under the curve is equivalent in every case—that is, we maintained an equivalent total synaptic input to every neuron in the HCO in every case, resulting in a conservation of synaptic weights. As a result of this normalization process, each neuron received the same total synaptic input as in the comparable homogeneous case. For example, a single neuron in a topologically symmetrical HCO (the homogeneous, or 18/18, case) in which $\Sigma\bar{g}_{\text{syn}-\text{i}} = 1$ nS received a

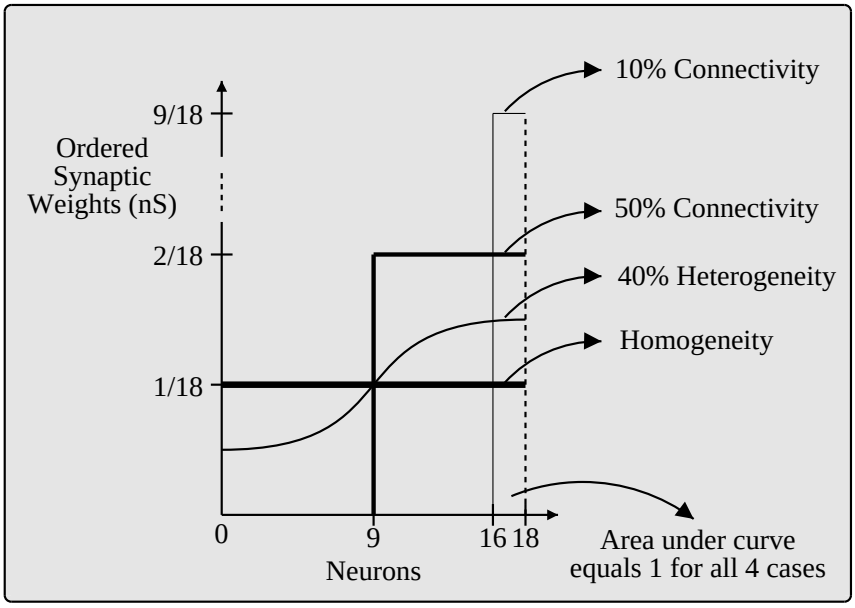


Figure 66: FPGA Synaptic-Weight Model.

synaptic input of $\Sigma\bar{g}_{\mathrm{syn-i}}/18 = 1/18$ nS from each of the 18 neurons on the contralateral side of the HCO. If topological heterogeneity was imposed by randomly disconnecting half of the contralateral synapses (the 50%, or 9/18, case), then each of the other nine synapses was required to have a value of $\Sigma\bar{g}_{\mathrm{syn-i}}/9 = 1/9$ nS to maintain consistency for purposes of comparison, resulting in a constant total synaptic input for both cases. As a result of this normalization process from the synaptic-weight model, we used $\Sigma\bar{g}_{\mathrm{syn-i}}$ and $\Sigma\bar{g}_{\mathrm{syn-e}}$ to indicate the *total* inhibitory and excitatory synaptic weight, respectively, applied to an individual neuron in the HCO.

Bursting. We defined *bursting* as a time trace of the membrane potential, V_{mem}, that met the following constraints (Section 2.3.2): (1) The number of spikes per burst was greater than 3. (2) The standard deviation of the peak voltages was less than 4 mV. (3) No bursting phase was four times as long as another bursting phase. (4) The period was less than 10 seconds. These constraints resulted in positive classifications for simple bursting (characterized by a single repeatable cycle of relatively constant period and duty cycle, consisting of a bursting phase followed by a silent phase) but not for complex bursting

(characterized by a single repeatable cycle consisting of multiple bursting and silent phases with a variety of durations or by an output in which the peak amplitudes were not of the same order).

Coherence. We defined *coherence* as the minimum percentage of neurons that were bursting on both sides of the HCO. Formally, coherence is given by

$$\text{Coherence} = \frac{\min(B1, B2)}{N/2} \times 100\% \tag{66}$$

where $B1$ ($B2$) is the number of bursting neurons in one (the other) half of the HCO and N is the total number of neurons in the network. This measure, unlike a simple summation of the number of bursting neurons, emphasized the rhythmic behavior that we were seeking. For example, if every neuron on one side of the HCO was bursting, but every neuron on the other side was not bursting, then the coherence was 0%, which represented no rhythmic activity, although $N/2$ neurons were bursting in the network. Alternatively, if half of the neurons on each side of the HCO were bursting (again resulting in $N/2$ neurons bursting in the network), then the coherence was 50%. This coherence measure also allowed us to plot bursting contours, such as 25%, 50%, and 75% contours. This measure was more physiologically relevant than a simple binary "yes–no" result that indicated whether all of the neurons in the network were bursting ("yes") or whether at least one of the neurons in the network was not bursting ("no") because not all of the neurons in a network are required to burst to obtain some behavior [79] [27]. In addition, the introduction of even slight parametric heterogeneity resulted in some neurons with relatively extreme parameter values, and these neurons would never be recruited to burst, causing "no" classifications for all cases.

Connectivity. We defined *connectivity*, or *connectedness*, as the percentage of non-zero synaptic connections between the inhibitory contralateral neurons. The design of the fully connected FPGA HCO provided complete control of the N^2 synaptic weights. A programmable weight of $\bar{g}_{\text{syn}-i,j} = 0$ nS implied a "broken" connection between neurons i and j. A dense (fully connected) network was defined as 100% connectivity, and the connectivity was smaller when a sparse set of network connections was tested, implying the

presence of topological heterogeneity as a result of the asymmetrical coupling.

Robustness. We defined *robustness*[1] in terms of the number of grid points that represent bursting activity in the burst-mapping diagram. Because it is a discrete number with a relatively low resolution (*i.e.*, the integration of the area has large differential areas), the number of grid points serves as an approximation for the area of the bursting space. We determined the grid sizes of each of the three burst-mapping diagrams by first determining the ranges of values in which bursting occurred for the homogeneous case and then dividing the parameter increment sizes such that a diagram included approximately $100 - 150$ grid points. We simply counted the grid points in the diagram and plotted this result against the coherence, resulting in a *cumulative distribution of coherence measures* that provided a *grid point–coherence slope*. To obtain these two-dimensional burst-mapping diagrams, we varied three intrinsic parameters ($\bar{g}_{\mathrm{NaP}}$, $\bar{g}_{\mathrm{Leak}}$, and E_{Leak}) from the 17 intrinsic parameters in the reduced model neuron. Therefore, we collected data for three burst-mapping diagrams from the $\frac{17 \cdot 16}{2} = 136$ possibilities: $\bar{g}_{\mathrm{NaP}} - E_{\mathrm{Leak}}$, $\bar{g}_{\mathrm{Leak}} - E_{\mathrm{Leak}}$, and $\bar{g}_{\mathrm{NaP}} - \bar{g}_{\mathrm{Leak}}$. We chose these three parameters for the following reasons: (1) Multiple parameters provided a broader result across parameter space. (2) E_{Leak} had more influence than any other parameter—a change in E_{Leak} affected all reversal potentials and half-maximal voltages because it determined the intrinsic baseline level of depolarization of bursting. (3) The $\bar{g}_{\mathrm{NaP}} - E_{\mathrm{Leak}}$ burst-mapping diagram was consistent with the published results [8]. (4) The critical subthreshold oscillations, which determined the ability of a neuron to intrinsically burst, were controlled by $\bar{g}_{\mathrm{NaP}}$ and $\bar{g}_{\mathrm{Leak}}$.

Two-dimensional burst-mapping diagrams were used because graphical information can often be difficult to understand and appreciate in three-dimensional form. We, however, showed families of curves for various $\bar{g}_{\mathrm{syn-i}}$ values and used multiple diagrams to show the various $\bar{g}_{\mathrm{syn-e}}$ values (shown later in Figure 68). We swept $\Sigma\bar{g}_{\mathrm{syn-i}}$ between $1-3$ nS in 1-nS increments and $\Sigma\bar{g}_{\mathrm{syn-e}}$ between $0 - 2$ nS in 1-nS increments. We chose these values for the following reasons: (1) The effects of an absence of ipsilateral excitatory coupling were tested by the condition $\Sigma\bar{g}_{\mathrm{syn-e}} = 0$ nS. (2) The values of $\Sigma\bar{g}_{\mathrm{syn-i}}$ and $\Sigma\bar{g}_{\mathrm{syn-e}}$ were of the

[1]Note that the term robustness, in addition to heterogeneity, is also not well defined in the literature.

same order as the critical subthreshold maximal conductances ($\bar{g}_{\text{Leak}}$ and $\bar{g}_{\text{NaP}}$). (3) The ratio of $\Sigma\bar{g}_{\text{syn}-\text{e}}$ and $\Sigma\bar{g}_{\text{syn}-\text{i}}$ was also consistent with (2).

We considered, but ruled out, other possibilities to determine the robustness of the bursting space. Researchers have previously considered such dynamical outputs as mean burst duration and mean burst frequency [9], but we were concerned about population-level dynamics [129]. For another possibility, the sum of the bursting ranges of all of the intrinsic parameters would have required the (arbitrary) addition of the disparate units mV, nS, and s. A similar problem would have existed if an analysis was performed to determine the most sensitive parameter. For example, the sensitivity analysis for a 0.1 nS increase in a maximal conductance parameter cannot be directly compared to the sensitivity analysis for a 0.1 ms increase in a time-constant parameter without the introduction of arbitrary rules. Even a simple sum of the bursting ranges of the maximal conductance parameters, which use the same units, would have been dominated by the one from $\bar{g}_{\text{Na}}$ because its bursting range was two orders of magnitude larger than the ranges of the other maximal conductances. Also, all of the half-maximal voltages (θ_x), sigmoidal-slope voltages (σ_x), and time-constant parameters (τ_x) were embedded in the calculations of the lookup tables of the FPGA and could not be changed easily. Although each version of the six lookup tables could have been implemented separately by using a parameterized multiplexor to select the appropriate output, this implementation would have required a significant number of additional resources, which would have reduced the network size, depending on how many parameter options were required.

Tests. The results of our tests are divided into three categories: preliminary, primary, and supplemental. The process that we used to obtain our results began with the *preliminary* tests, in which we crudely explored the parameter space to determine appropriate sizes of the grid points and the parameter ranges for each of the three burst-mapping diagrams. The *primary* tests were designed to impose heterogeneity in the network in a variety of forms and in a variety of intrinsic and synaptic parameter spaces. We categorized those tests as *supplemental* that were extensions of the primary tests and in which less data was required. Table 18 shows the test summaries, which includes the test numbers,

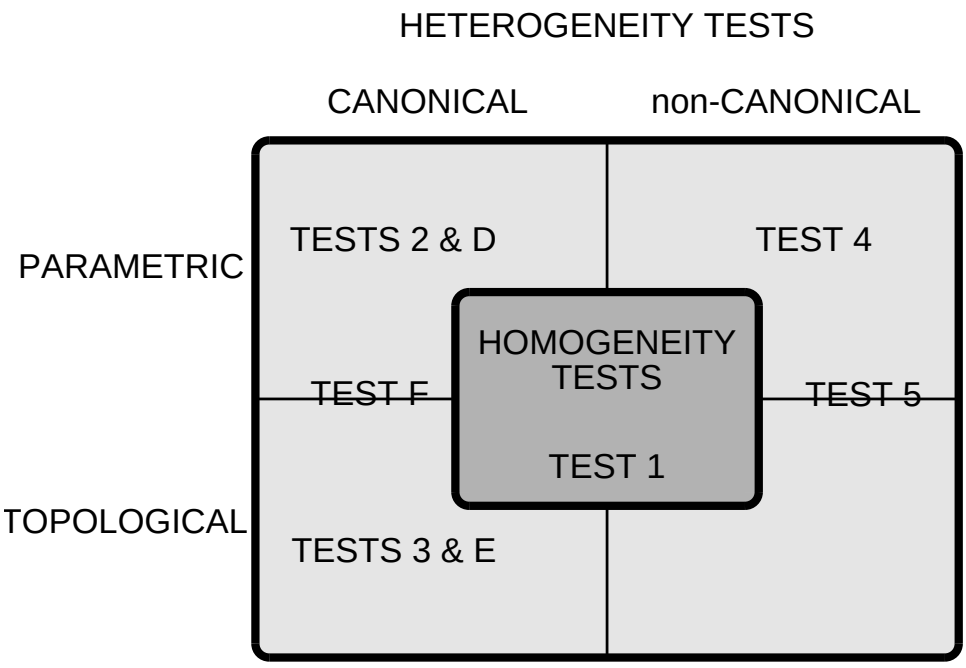


Figure 67: Chapter 4 summary—relationships between the tests.

short descriptions of each test including the network configurations, and the corresponding figure numbers. A graphical classification of the tests is also shown in Figure 67, in which their locations in the heterogeneity "spectrum" are indicated. Each of the primary tests required 27 two-dimensional burst-mapping diagrams for the nine different combinations of inhibitory and excitatory synaptic weights (Figure 68), and the grid definitions of each of the three burst-mapping diagrams are shown in Table 19. A pictorial of the presentation of the data for the primary tests is shown in Figure 69. On the left, the number of grid points as a function of the coherence measure for a fixed $\Sigma\bar{g}_{\mathrm{syn-e}}$ is shown, termed the *cumulative distribution of coherence measures*, resulting in the *grid point–coherence slope*, and the size of the bursting space for an isolated neuron is shown by the horizontal dashed line. The corresponding burst-mapping diagrams are shown on the right for 50% coherence, and for all of the plots in the figure, the darker the line, the larger the $\Sigma\bar{g}_{\mathrm{syn-i}}$.

4.2 Results

The goal of our work was to use a network of neurons to study the role of heterogeneity in a variety of forms, in a variety of intrinsic parameter spaces, and in a variety of inhibitory and

Table 18: Chapter 4 summary—tests, figures, and network configurations.

Test	Figures	Description
		Preliminary Tests
A	70–71	• justification of the coherence measure
B	72	• analysis of an isolated neuron
C	73	• analysis of the network size
		Primary Tests
1	74–76	• homogeneous configuration for $N = 36$ • equivalent intrinsic and synaptic parameters
2	77–79	• 40% heterogeneity in the intrinsic parameters • canonical point: $\bar{g}_{\mathrm{NaP}} = 2.8$ nS, $\bar{g}_{\mathrm{Leak}} = 2.8$ nS, $E_{\mathrm{Leak}} = -60$ mV
3	82–84	• sparseness in the contralateral inhibitory synaptic connections • 50% of connections have double synaptic strength (random)
4	92–94	• "non-canonical point" with parametric heterogeneity • non-canonical point: $\bar{g}_{\mathrm{NaP}} = 4.0$ nS, $\bar{g}_{\mathrm{Leak}} = 2.0$ nS, $E_{\mathrm{Leak}} = -59$ mV • 40% heterogeneity in the intrinsic parameters
5	95–97	• "non-canonical point" with parametric and topological heterogeneity • non-canonical point: $\bar{g}_{\mathrm{NaP}} = 4.0$ nS, $\bar{g}_{\mathrm{Leak}} = 2.0$ nS, $E_{\mathrm{Leak}} = -59$ mV • 40% heterogeneity in the intrinsic parameters • 50% of connections have double synaptic strength (random)
		Supplemental Tests
D	80–81	• extension of Test 2 for $\bar{g}_{\mathrm{NaP}} - E_{\mathrm{Leak}}$ diagram only • 20% and 60% heterogeneity cases
E	85–87	• extension of Test 3 for $\bar{g}_{\mathrm{NaP}} - E_{\mathrm{Leak}}$ diagram only • sparse connectivity configurations for 4/18, 2/18, 1/18 probabilities
F	88–91	• extension of Test 2 and Test 3 for $\bar{g}_{\mathrm{NaP}} - E_{\mathrm{Leak}}$ diagram only • sparse connectivity configurations for 9/18, 4/18, 2/18, 1/18 probabilities and 40% heterogeneity

Table 19: Chapter 4 summary—formats of the burst-mapping diagrams for the primary tests. The units of $\bar{g}_{\mathrm{NaP}}$ and $\bar{g}_{\mathrm{Leak}}$ are nS, and the unit of E_{Leak} is mV.

2-D Space	Grid Point Size	Parameter Sweeps	Points
$\bar{g}_{\mathrm{NaP}} - \bar{g}_{\mathrm{Leak}}$	1.0×1.0	$\bar{g}_{\mathrm{NaP}} = [1:1:20]$, $\bar{g}_{\mathrm{Leak}} = [1:1:8]$	160
$\bar{g}_{\mathrm{NaP}} - E_{\mathrm{Leak}}$	2.0×2.5	$\bar{g}_{\mathrm{NaP}} = [1:2:17]$, $E_{\mathrm{Leak}} = [-67:2.5:-42]$	99
$\bar{g}_{\mathrm{Leak}} - E_{\mathrm{Leak}}$	1.0×2.0	$\bar{g}_{\mathrm{Leak}} = [1:1:10]$, $E_{\mathrm{Leak}} = [-68:2:-42]$	140

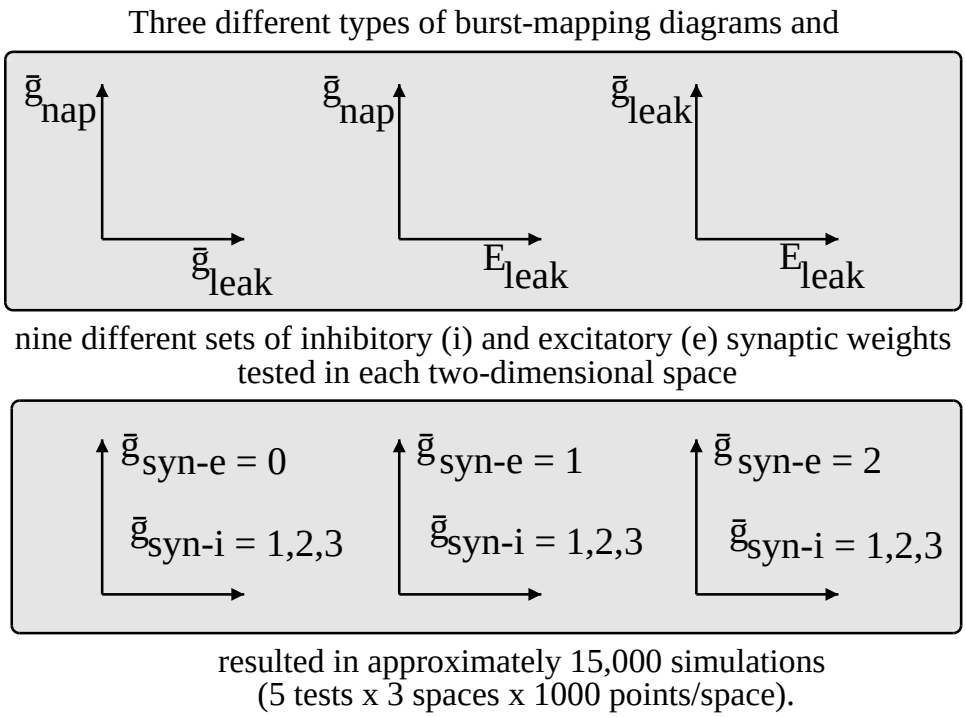


Figure 68: Chapter 4 summary—format of the primary test results (Part I).

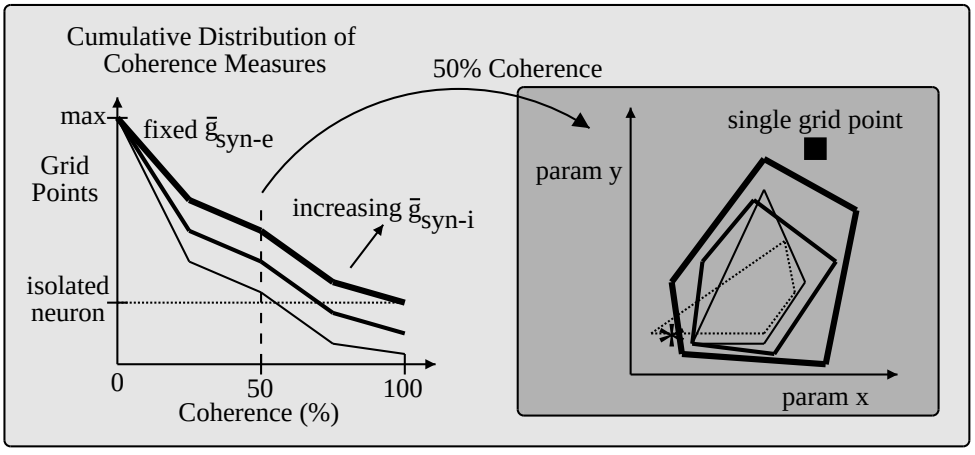


Figure 69: Chapter 4 summary—format of the primary test results (Part II).

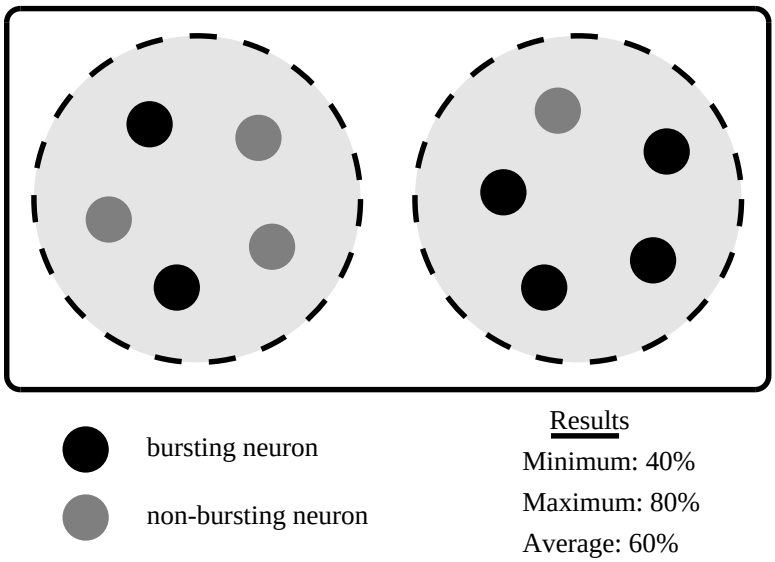


Figure 70: FPGA preliminary Test A—measures of network dynamics. Some of the possibilities of a coherence measure include the minimum, maximum, and average (or total) number of bursting neurons.

excitatory synaptic-weight configurations. To do so, we began with a few preliminary tests that provided the foundation for our more extensive primary tests, which were continued with our supplemental tests.

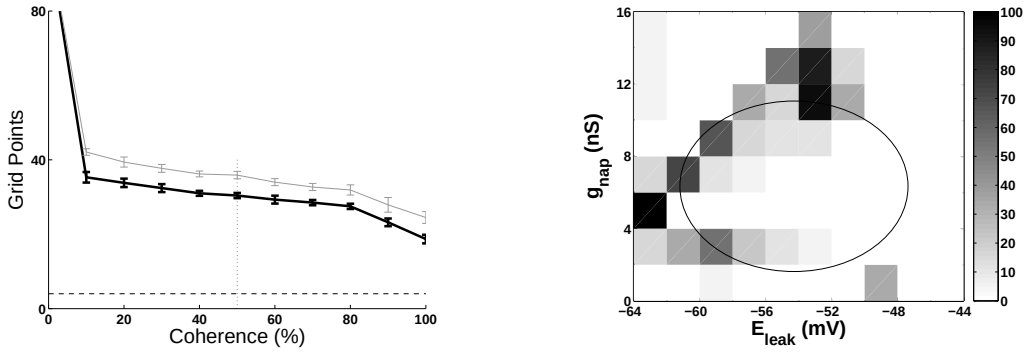
4.2.1 Preliminary Tests

Preliminary Test A—Justification of the Coherence Measure. We previously defined *coherence* as the minimum percentage of neurons that were bursting on either side of the HCO. This measure, however, did not use the information from the other side of the HCO that possibly had additional bursting neurons. Alternatively, we could have defined the coherence measure as the maximum percentage of neurons bursting on either side or the total percentage of neurons bursting in the HCO (Figure 70), but the maximum measure would also suffer from a similar loss of data as the minimum measure. Our use of minimum, however, more accurately reflected the objective of our work—the study of rhythmic movement—as explained previously in the *Coherence* part of Section 4.1.

We determined the stability of the *cumulative distribution of coherence measures* by performing a specific test ten times and also determined where the variability of this distribution was most likely to occur. Figure 71 shows the data for a specific trial from primary

test 2: 40% heterogeneity applied to $\bar{g}_{\text{Leak}}$, $\Sigma\bar{g}_{\text{syn}-\text{i}} = 3$ nS, and $\Sigma\bar{g}_{\text{syn}-\text{e}} = 2$ nS. Figure 71A shows that the cumulative distribution measures, using either a minimum or maximum definition, are both stable and closely related, and it also shows the *grid point–coherence slope*. Figure 71B shows, as expected, that most of the variance in the minimum coherence measure occurs at the edge of the burst region. Although small, the differences between the ten tests were a manifestation of the random nature of the experiment—that is, heterogeneity produced non-deterministic results, as expected. Therefore, in our subsequent analysis in this chapter, we were careful not to take a strict reading of the number of grid points when making a comparison between the sizes of bursting regions because we expected the cumulative distribution of coherence measures to fluctuate similarly to that shown in Figure 71A. In addition, the plots that were used with our primary tests, which were in the format of Figure 71A, generally do not show crossover between the $\Sigma\bar{g}_{\text{syn}-\text{i}} = 1, 2, 3$ nS data, which is another validation of the minimum coherence measure. The stability of the data was also a result of how we conducted this specific test (and all of our heterogeneity tests in this chapter)—for every grid point, a new random set of $\bar{g}_{\text{Leak}}$ values was generated, resulting in an averaging effect over the entire grid, rather than a biased effect resulting from a single set of values that were used repeatedly. As a result of these findings and our methods, we were confident in the use of the *minimum* percentage of neurons that were bursting on either side of the HCO as a stable metric to determine population-based bursting statistics, and this measure of *coherence* was therefore used in our subsequent analysis.

Preliminary Test B—Analysis of an Isolated Neuron. To determine a baseline size for all of the burst-mapping regions shown in this chapter, we examined the changes to the three burst-mapping diagrams for an isolated neuron as we swept the values of the three intrinsic parameters of interest—$\bar{g}_{\text{Leak}}$, $\bar{g}_{\text{NaP}}$, and E_{Leak}. The three diagrams are shown in Figure 72, and background information related to these three tests is shown in Table 20. Although our subsequent testing focused on the points in parameter space that were closer to the canonical areas, we used wide sweeps for each of these subfigures to show the nature of the continuous changes of the bursting regions. These continuous changes are indicated in all three subfigures by the overlapping regions and would be more pronounced if we



A. B.

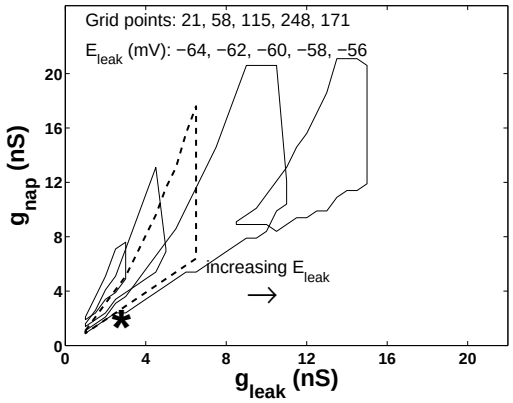
Figure 71: FPGA preliminary Test A—an analysis of the stability of the cumulative distribution of coherence measures for a specific trial from primary test 2: 40% heterogeneity applied to $\bar{g}_{\text{Leak}}$, $\Sigma\bar{g}_{\text{syn-i}} = 3$ nS, and $\Sigma\bar{g}_{\text{syn-e}} = 2$ nS. A. Coherence measure (minimum) is the heavy line, and the maximum measure is the light line. The error bars indicate the standard deviation after ten trials. B. The range of the minimum coherence measure for each of the grid points over all ten trials. The maximum is 100 and implies that the coherence was 0 for at least one trial and 100 for at least one trial at a specific grid point. The oval indicates the approximate location of the bursting region.

Table 20: FPGA preliminary Test B—background information for Figure 72.

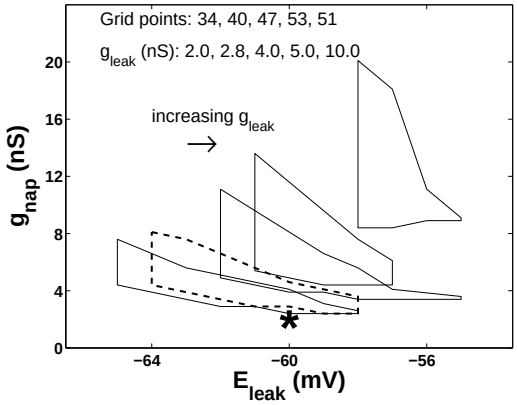
Figure	2-D Space	Grid Point Size	Parameter Sweep
Figure 72C	$\bar{g}_{\text{NaP}} - \bar{g}_{\text{Leak}}$	0.5 nS × 0.5 nS	$E_{\text{Leak}} = -64, -62, -60, -58, -56$ mV
Figure 72A	$\bar{g}_{\text{NaP}} - E_{\text{Leak}}$	0.5 nS × 1 mV	$\bar{g}_{\text{Leak}} = 2.0, 2.8, 4.0, 5.0, 10.0$ nS
Figure 72B	$\bar{g}_{\text{Leak}} - E_{\text{Leak}}$	0.5 nS × 1 mV	$\bar{g}_{\text{NaP}} = 2.0, 2.8, 4.0, 5.2, 10.0$ nS

obtained higher-resolution data. The burst-mapping diagrams using the canonical values of the parameters (the dashed contours) will be used as baseline diagrams in later figures. Because $\bar{g}_{\text{NaP}}$ had a larger bursting range than $\bar{g}_{\text{Leak}}$, the burst-mapping diagrams are expectedly larger in Figure 72B than in Figure 72C (the ranges of both plots are the same). The following two important findings will be more fully evaluated with primary test 4 and primary test 5, in which the tests were completed at a non-canonical point in parameter space: (1) In all three subfigures, the canonical parameter values are on the edges of the burst-mapping diagrams. (2) In all three subfigures, the bursting region using the canonical parameter value is not the largest one.

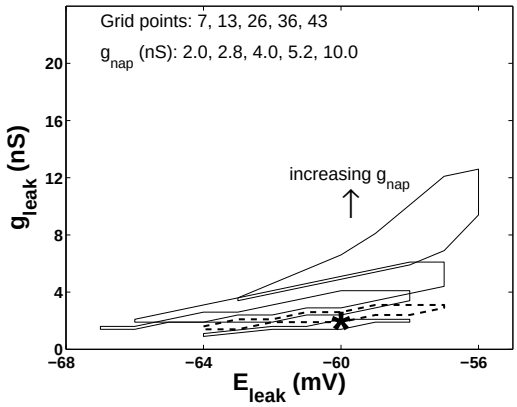
Preliminary Test C—Analysis of the Network Size. To determine how the number of neurons in the network, N, affected our results, we analyzed a particular point in synaptic parameter space for a variety of network sizes using the homogeneous configuration and also



A.



B.

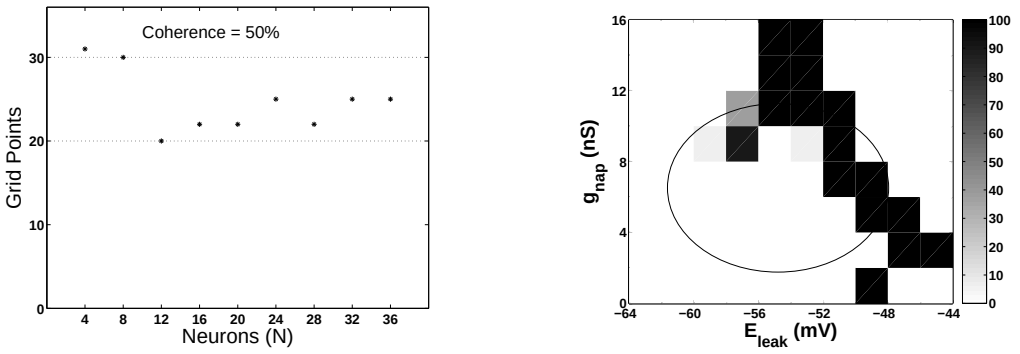


C.

Figure 72: FPGA preliminary Test B—burst-mapping diagrams for the isolated neuron. The canonical parameter values are indicated by the * in each of the plots. In each subfigure, the dashed region represents the burst-mapping diagram for the canonical value of the parameter that is being swept. The title of each subfigure includes the number of grid points, or area, of each of the burst-mapping diagrams, in increasing order of the parameter value. The x-axis and y-axis ranges in the B. and C. plots are the same.

determined where the variability was most likely to occur. Figure 73 shows the results for $N = 4 - 36$, in increments of four. We chose the same arbitrary test point from preliminary test A: $\bar{g}_{\mathrm{NaP}} - E_{\mathrm{Leak}}$ space for $\Sigma\bar{g}_{\mathrm{syn-i}} = 3$ nS and $\Sigma\bar{g}_{\mathrm{syn-e}} = 2$ nS. The normalization of the synaptic weights to the size of the network was necessary to decouple size effects as explained previously in the *Synaptic-Weight Model* part of Section 4.1. Note that we could not perform similar heterogeneity tests because the application of heterogeneity to a network of size, say, $N = 4$ was not possible, by definition. Figure 73A shows that the sizes of the bursting regions in terms of grid points for 50% coherence vary between 20 and 31, but not in a systematic way (the total number of grid points in the $\bar{g}_{\mathrm{NaP}} - E_{\mathrm{Leak}}$ space was 99). Since these are results from homogeneous configurations, the results are largely independent of the coherence value since, as will be shown later, an all-or-nothing result generally occurs for homogeneous configurations, and even more so as the size of the network is reduced. Figure 73B shows, as expected, that most of the variance in the coherence measure occurs at the edge of the burst region. The variance of the grid points shown in Figure 73A was larger than expected, but more importantly, the origin of the seemingly random results was unknown; our best interpretation was that the variation was possibly another manifestation of the quantization error of the FPGA model, which was discussed in Section 2.3.5.

We also extended this analysis by recording single-neuron anti-phasic bursting characteristics for an individual neuron in a variety of network sizes using the canonical set of intrinsic parameters for a given total inhibitory input ($\Sigma\bar{g}_{\mathrm{syn-i}}$) and a given total excitatory input ($\Sigma\bar{g}_{\mathrm{syn-e}}$). As shown in Table 21, these single-neuron bursting characteristics are similar and relatively independent of the number of neurons in the network . Therefore, a neuron in a network of size N that receives $N/2$ inhibitory connections (from the contralateral half of the HCO) totaling $\Sigma\bar{g}_{\mathrm{syn-i}}$ and $N/2 - 1$ excitatory connections (from the ipsilateral half of the HCO) totaling $\Sigma\bar{g}_{\mathrm{syn-e}}$ will have similar anti-phasic bursting dynamics as a neuron in a network with a different number of neurons, controlling for $\Sigma\bar{g}_{\mathrm{syn-i}}$ and $\Sigma\bar{g}_{\mathrm{syn-e}}$ (*i.e.*, normalized for the network size). This constant level of input to each neuron decoupled network size effects and is an important model to follow to obtain comparable



A. B.

Figure 73: FPGA preliminary Test C—an analysis of the network size. A. Size of the bursting region in terms of grid points for 50% coherence for networks of size $N = 4 - 36$, in increments of four. B. The range of the coherence measure for each of the grid points for all network sizes. The maximum is 100 and implies that the coherence was 0 for at least one network size and 100 for at least one other network size at a specific grid point. The oval indicates the approximate location of the bursting region.

Table 21: FPGA preliminary Test C—single-neuron bursting characteristics for various network configurations using the canonical set of intrinsic parameters and a variety of synaptic weights. The duty cycles were $8 - 11\%$ for all cases.

Synaptic Weights (nS)		Period (s)		Spike Freq (Hz)		Spikes/Burst	
$\Sigma\bar{g}_{\mathrm{syn-i}}$	$\Sigma\bar{g}_{\mathrm{syn-e}}$	$N = 4, 8$	$N = 16, 36$	$4, 8$	$16, 36$	$4, 8$	$16, 36$
1,2,3	1	$7.7 - 8.0$	$7.0 - 7.3$	$48 - 49$	$50 - 52$	$40 - 41$	$35 - 36$
1,2,3	2	$8.7 - 9.1$	$8.0 - 8.4$	$61 - 63$	$69 - 70$	$56 - 58$	$49 - 51$

results with other configurations. As a result of this preliminary test in which we found the bursting characteristics to be relatively independent of N, we did not consider N to be a network parameter. We used a network size of $N = 36$ for all subsequent testing.

4.2.2 Primary Test 1—Homogeneous Configuration

The most important goal of our work was to determine whether heterogeneity increased the robustness of a network of neurons, and this measure of robustness was determined by the sizes of three bursting regions. To perform this analysis, we were required to compare the results from a heterogeneous configuration to those from a *control*. The appropriate control for our testing is the homogeneous configuration. In other words, if a form of heterogeneity that is imposed on the network in some region of intrinsic and synaptic parameter space is deemed to be functionally advantageous, then it must result in a bursting region that is

larger than the one that would result without heterogeneity—that is, from the homogeneous configuration.

For the homogenous configuration, every neuron had identical intrinsic parameters and made identical excitatory and inhibitory synaptic connections—that is, for each half of the HCO, every neuron was identical and interchangeable. The results of the homogeneity tests are shown in Figures 74–76. Because the neuron population was homogeneous, when complete network rhythmicity did occur, all neurons on a given side of the HCO had similar single-neuron bursting characteristics, resulting in a *characteristic neuron.* As a result, such measures as the periods, duty cycles, and average spike frequencies were approximately equivalent, but as an intrinsic parameter or a synaptic weight changed, the dynamics of the characteristic neuron also changed in response.

We first examined how the bursting range of $\bar{g}_{\mathrm{NaP}}$ changed as a result of the neuron being synaptically connected versus being isolated. The bursting range of $\bar{g}_{\mathrm{NaP}}$ for the isolated neuron was $2.7 - 4.9$ nS. For the coupled neuron in a homogeneous network, the bursting range was significantly larger; for example, the bursting range of $\bar{g}_{\mathrm{NaP}}$ for $\Sigma\bar{g}_{\mathrm{syn-i}} = 2.0$ nS and $\Sigma\bar{g}_{\mathrm{syn-e}} = 0$ nS was $3 - 16$ nS. As a result, a general finding about homogeneous networks was that they worked, or were advantageous, in the sense that the bursting region of a coupled neuron in a homogeneous network was larger than the bursting region of an uncoupled neuron, for the levels of synaptic coupling that were tested.

We also examined the roles of the excitatory and inhibitory synaptic connections in this homogeneous network, but the results were inconclusive for both types of connections. The results for the inhibitory connections from Figure 74 show that for $\Sigma\bar{g}_{\mathrm{syn-e}} = 0 - 1$ nS, the size of the bursting region increases as $\Sigma\bar{g}_{\mathrm{syn-i}}$ goes from $1 - 2$ nS and then decreases from $2 - 3$ nS, which is not the case for $\Sigma\bar{g}_{\mathrm{syn-e}} = 2$ nS as the size of the bursting region reduces for $\Sigma\bar{g}_{\mathrm{syn-i}}$ from $1 - 3$ nS. In addition, in the $\bar{g}_{\mathrm{NaP}} - \bar{g}_{\mathrm{Leak}}$ space, excitatory coupling tends to decrease the size of the bursting region, but that is not the case in the $\bar{g}_{\mathrm{NaP}} - E_{\mathrm{Leak}}$ space. As a result, for the homogeneous network, the addition of excitatory coupling will not necessarily increase the size of the bursting space. Excitatory coupling would be beneficial when bursting neurons recruit non-bursting neurons, but if all of the neurons in

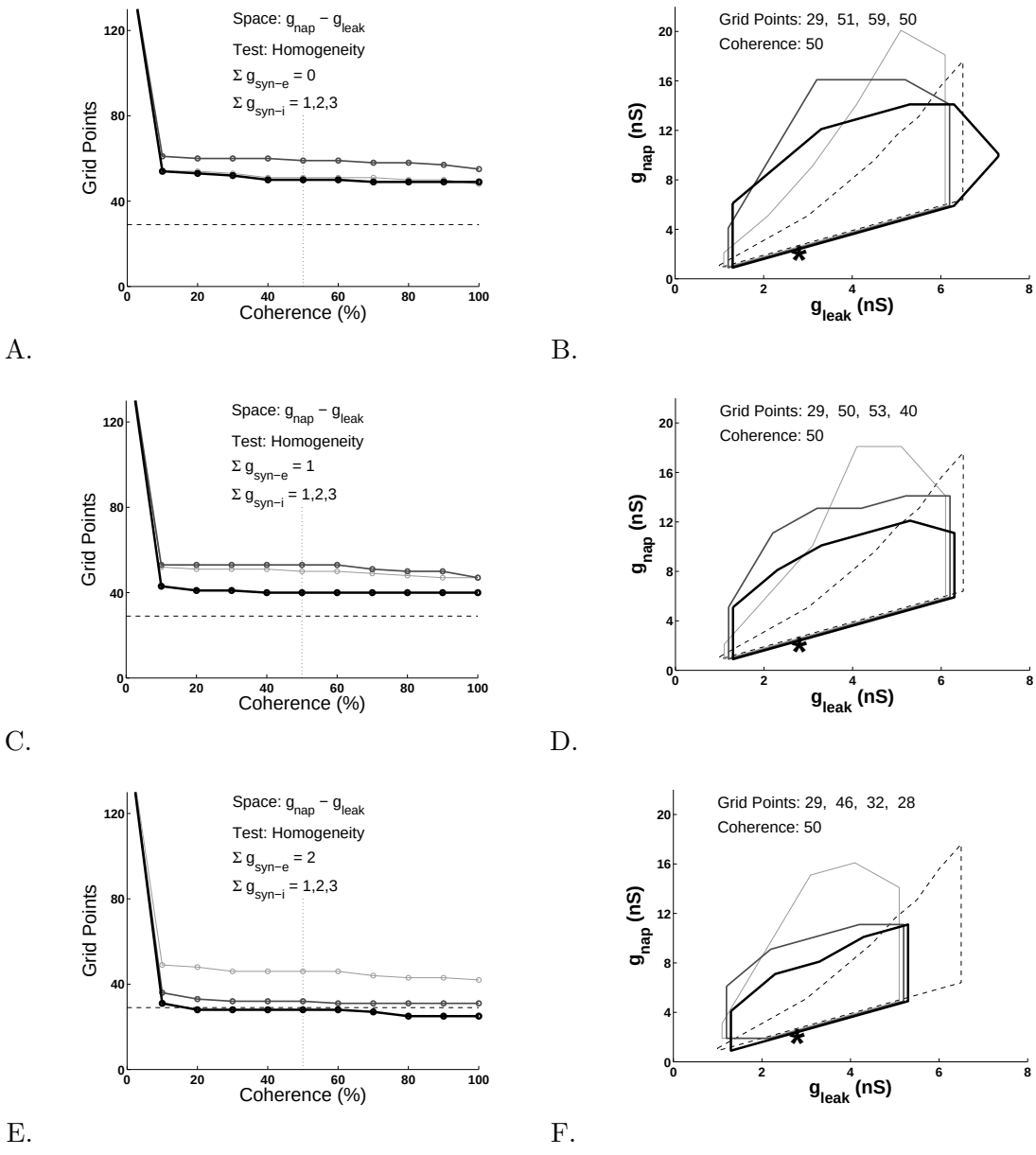
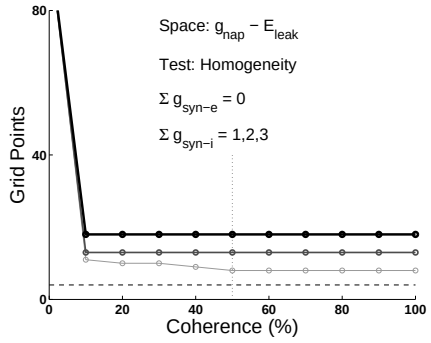
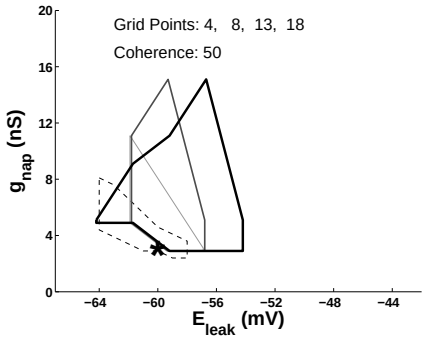


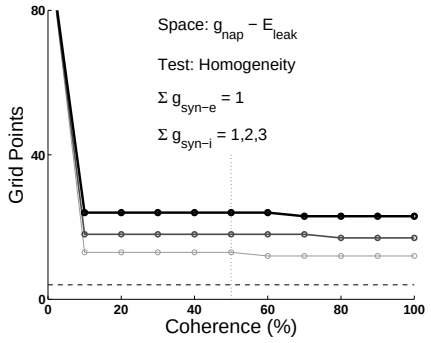
Figure 74: FPGA primary Test 1—$\bar{g}_{\mathrm{NaP}} - \bar{g}_{\mathrm{Leak}}$ space for the homogeneous configuration. Figures A, C, and E show the cumulative distribution of coherence measures for increasing values of ipsilateral excitation ($\Sigma\bar{g}_{\mathrm{syn-e}}$) and contralateral inhibition ($\Sigma\bar{g}_{\mathrm{syn-i}}$). The 0 percent coherence level represents every grid point (160). The horizontal dotted lines represent the case for the isolated neuron (29). The vertical dashed lines indicate the 50 percent coherence level, and the corresponding burst-mapping diagrams are shown in figures B, D, and F.



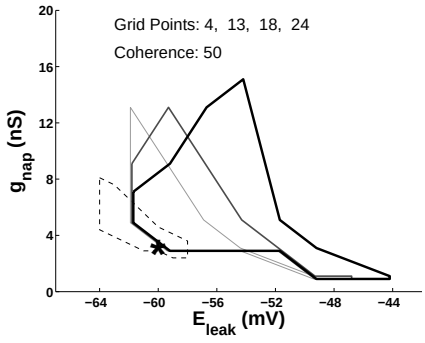
A.



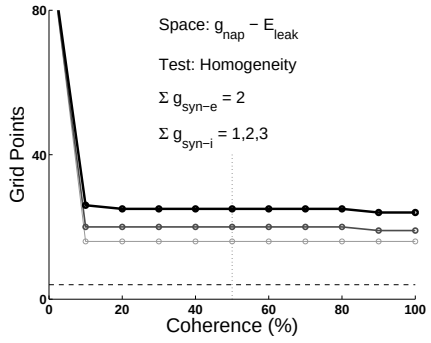
B.



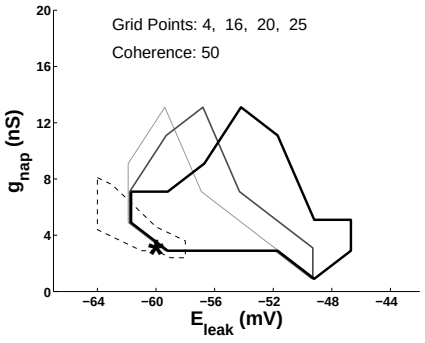
C.



D.



E.



F.

Figure 75: FPGA primary Test 1—$\bar{g}_{\mathrm{NaP}} - E_{\mathrm{Leak}}$ space for the homogeneous configuration. Figures A, C, and E show the cumulative distribution of coherence measures for increasing values of ipsilateral excitation ($\Sigma\bar{g}_{\mathrm{syn-e}}$) and contralateral inhibition ($\Sigma\bar{g}_{\mathrm{syn-i}}$). The 0 percent coherence level represents every grid point (99). The horizontal dotted lines represent the case for the isolated neuron (4). The vertical dashed lines indicate the 50 percent coherence level, and the corresponding burst-mapping diagrams are shown in figures B, D, and F.

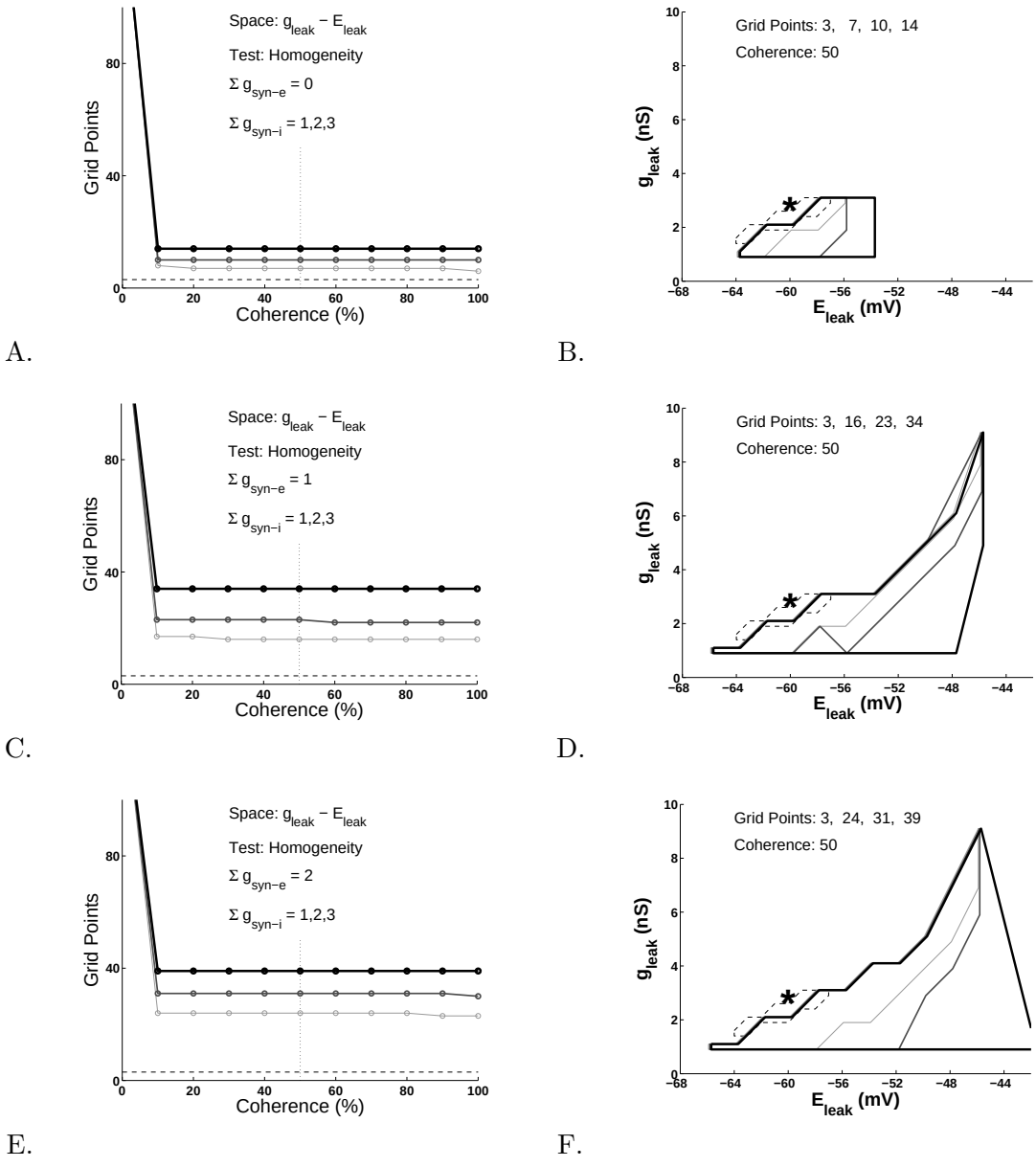


Figure 76: FPGA primary Test 1—$\bar{g}_{\mathrm{Leak}} - E_{\mathrm{Leak}}$ space for the homogeneous configuration. Figures A, C, and E show the cumulative distribution of coherence measures for increasing values of ipsilateral excitation ($\Sigma\bar{g}_{\mathrm{syn-e}}$) and contralateral inhibition ($\Sigma\bar{g}_{\mathrm{syn-i}}$). The 0 percent coherence level represents every grid point (140). The horizontal dotted lines represent the case for the isolated neuron (3). The vertical dashed lines indicate the 50 percent coherence level, and the corresponding burst-mapping diagrams are shown in figures B, D, and F.

a homogeneous network were exhibiting the same behavior, then the result would be the single characteristic neuron.

Although the burst-mapping diagrams were constructed for coherence measures of 50% (partial network rhythmicity), approximately 98% of the grid points in the figures resulted in coherence measures of 0% or 100% (the other 2% of the grid points were on the edges of the diagrams, which is the expected location of the variance), and therefore burst-mapping diagrams for coherence measures of, say, 25% or 75% would not have differed appreciably. As a result of the grid point–coherence slope that is virtually zero, a homogeneous configuration will result in a burst-mapping space in which one of the following generally occurs: (1) all neurons in the HCO are bursting and exhibit perfect frequency locking or (2) all neurons in at least one half of the HCO are not bursting.[2] This is a general *all-or-nothing* characteristic of homogeneous networks and will be compared to the *graded* result that is characteristic of heterogeneous networks, which will be described next.

4.2.3 Primary Test 2—40% Heterogeneity in the Intrinsic Parameters

In this first study of heterogeneity, we were interested in testing the role of intrinsic parametric heterogeneity in a network of neurons. For our first set of tests, we applied 40% heterogeneity to the three intrinsic parameters of interest—$\bar{g}_{\mathrm{NaP}}$, $\bar{g}_{\mathrm{Leak}}$, and E_{Leak}. As previously discussed in the *Heterogeneity* part of Section 4.1, the application of 40% heterogeneity to a maximal conductance resulted in a normal distribution of parameter values that had a coefficient of variation of 40%; for E_{Leak}, that corresponded to a coefficient of variation of 3%. After preliminary testing, we chose 40% as a good starting point for our heterogeneity tests because of the distinctly different, but not wholly unrelated, results compared to those from the homogeneous case. We complemented this study with supplemental test D in which 20% and 60% heterogeneity was applied to $\bar{g}_{\mathrm{Leak}}$ only.

The results of the tests for 40% heterogeneity are shown in Figures 77–79. Like the homogeneous case, the roles of the inhibitory and excitatory coupling were mixed: In the $\bar{g}_{\mathrm{NaP}} - E_{\mathrm{Leak}}$ and $\bar{g}_{\mathrm{Leak}} - E_{\mathrm{Leak}}$ spaces, for the level of synaptic input tested, the larger

[2]For case (2), when complete network rhythmicity did not occur, the neuronal outputs on one side of the HCO did not necessarily look identical, although a symmetry was evident.

the inhibitory (excitatory) synaptic weight, the larger the bursting space for a given level of excitatory (inhibitory) input. These were not the results, however, in the $\bar{g}_{\mathrm{NaP}} - \bar{g}_{\mathrm{Leak}}$ space.

The most conclusive result from this set of tests was that if 100% coherence (or nearly 100% coherence) was required, then the introduction of heterogeneity would not benefit the network—that is, homogeneity would be required. In fact, the size of the bursting region in $\bar{g}_{\mathrm{NaP}} - \bar{g}_{\mathrm{Leak}}$ space, for all values of the synaptic weights tested, was smaller than the one for the isolated neuron for coherence values of 90% or greater. The next result was given by the steepness of the grid point–coherence slope: the tradeoff with the small bursting regions for large coherence measures was the large bursting regions for small coherence measures, in which the coherence measure played a large role in determining the size of the bursting region. Therefore, heterogeneity resulted in a bursting region that could be larger than the one in the homogeneous case or smaller than the one in the isolated case, depending on the coherence measure. As a result of the similar locations of the 10% and 90% coherence points on the cumulative distribution of coherence measures plots, the sizes of the bursting regions for the 50% coherence measure with heterogeneity was approximately the same as the sizes of the bursting regions with no heterogeneity.

Supplemental Test D—20% and 60% Heterogeneity in $\bar{g}_{\mathrm{Leak}}$. To continue the analysis of the role of intrinsic parametric heterogeneity, we also imposed 20% (Figure 80) and 60% (Figure 81) heterogeneity to $\bar{g}_{\mathrm{Leak}}$ only. As expected, the grid point–coherence slope increased monotonically as the heterogeneity increased—that is, the grid point–coherence slope was qualitatively[3] proportional to heterogeneity. In addition, the sizes of the bursting regions for each of the cases for 50% coherence was approximately the same (Table 22). The results for the 60% heterogeneity case, however, showed a slight trend toward smaller bursting regions, and this gives another general result: although heterogeneity can be advantageous, particularly for low values of coherence, at some level of heterogeneity the benefits must begin to be reduced. We did not determine the level of heterogeneity in which virtually all benefits of the heterogeneity were eliminated (*i.e.*, where the size of a

[3]We did not quantify the value of the grid point–coherence slope.

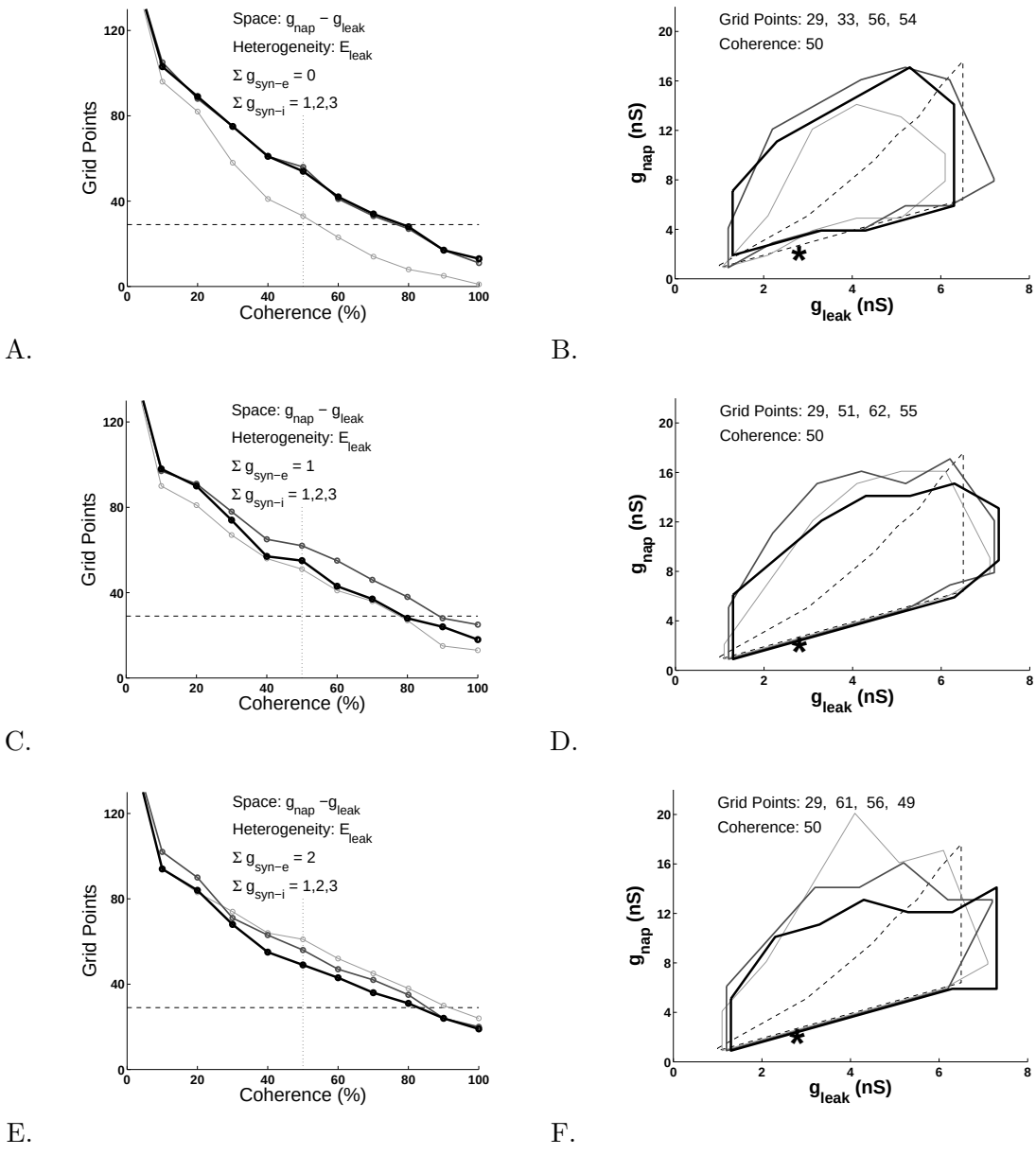


Figure 77: FPGA primary Test 2—$\bar{g}_{\mathrm{NaP}} - \bar{g}_{\mathrm{Leak}}$ space for 40% intrinsic heterogeneity. Figures A, C, and E show the cumulative distribution of coherence measures for increasing values of ipsilateral excitation ($\Sigma\bar{g}_{\mathrm{syn-e}}$) and contralateral inhibition ($\Sigma\bar{g}_{\mathrm{syn-i}}$). The 0 percent coherence level represents every grid point (160). The horizontal dotted lines represent the case for the isolated neuron (29). The vertical dashed lines indicate the 50 percent coherence level, and the corresponding burst-mapping diagrams are shown in figures B, D, and F.

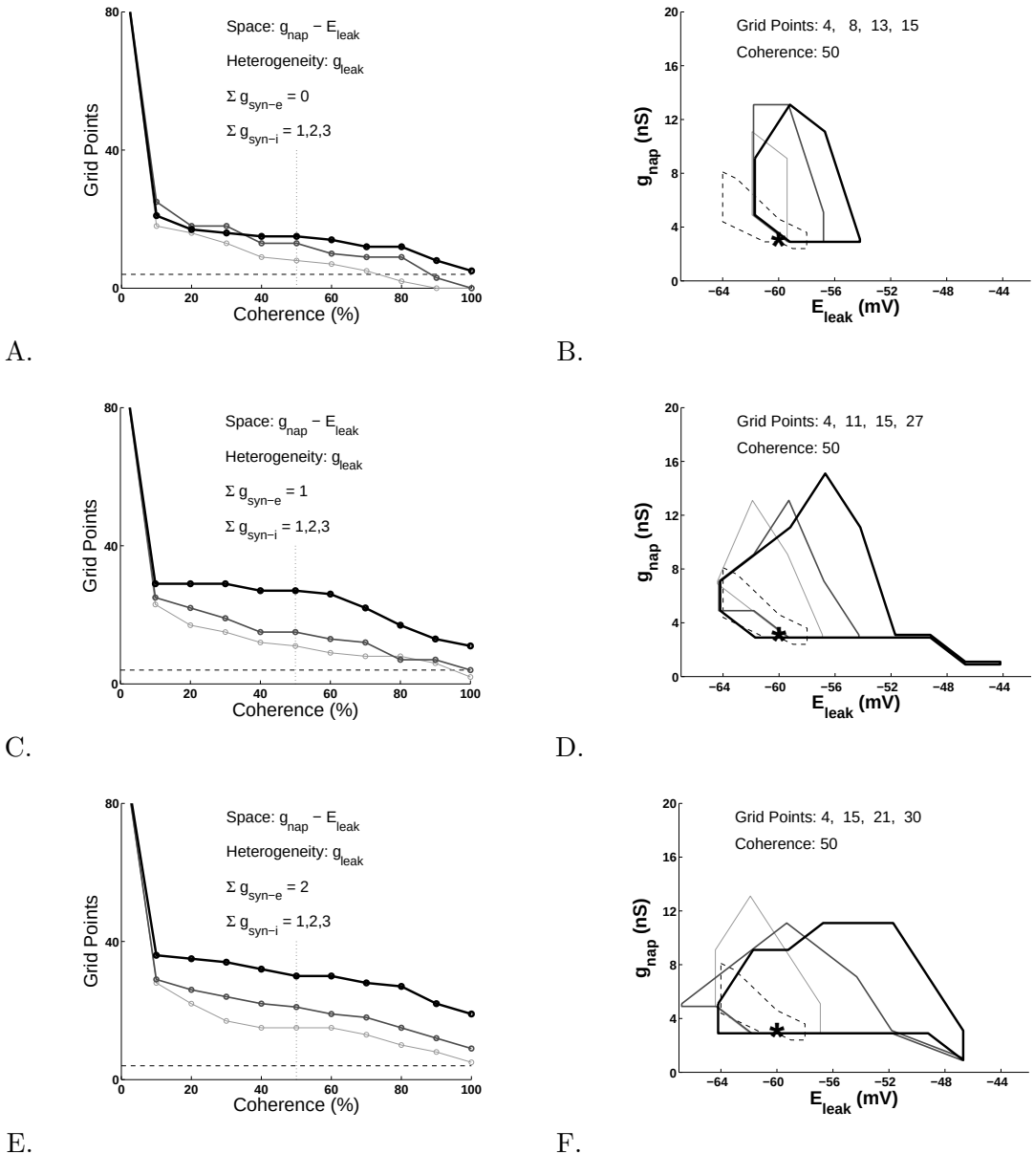


Figure 78: FPGA primary Test 2—$\bar{g}_{\mathrm{NaP}} - E_{\mathrm{Leak}}$ space for 40% intrinsic heterogeneity. Figures A, C, and E show the cumulative distribution of coherence measures for increasing values of ipsilateral excitation ($\Sigma\bar{g}_{\mathrm{syn-e}}$) and contralateral inhibition ($\Sigma\bar{g}_{\mathrm{syn-i}}$). The 0 percent coherence level represents every grid point (99). The horizontal dotted lines represent the case for the isolated neuron (4). The vertical dashed lines indicate the 50 percent coherence level, and the corresponding burst-mapping diagrams are shown in figures B, D, and F.

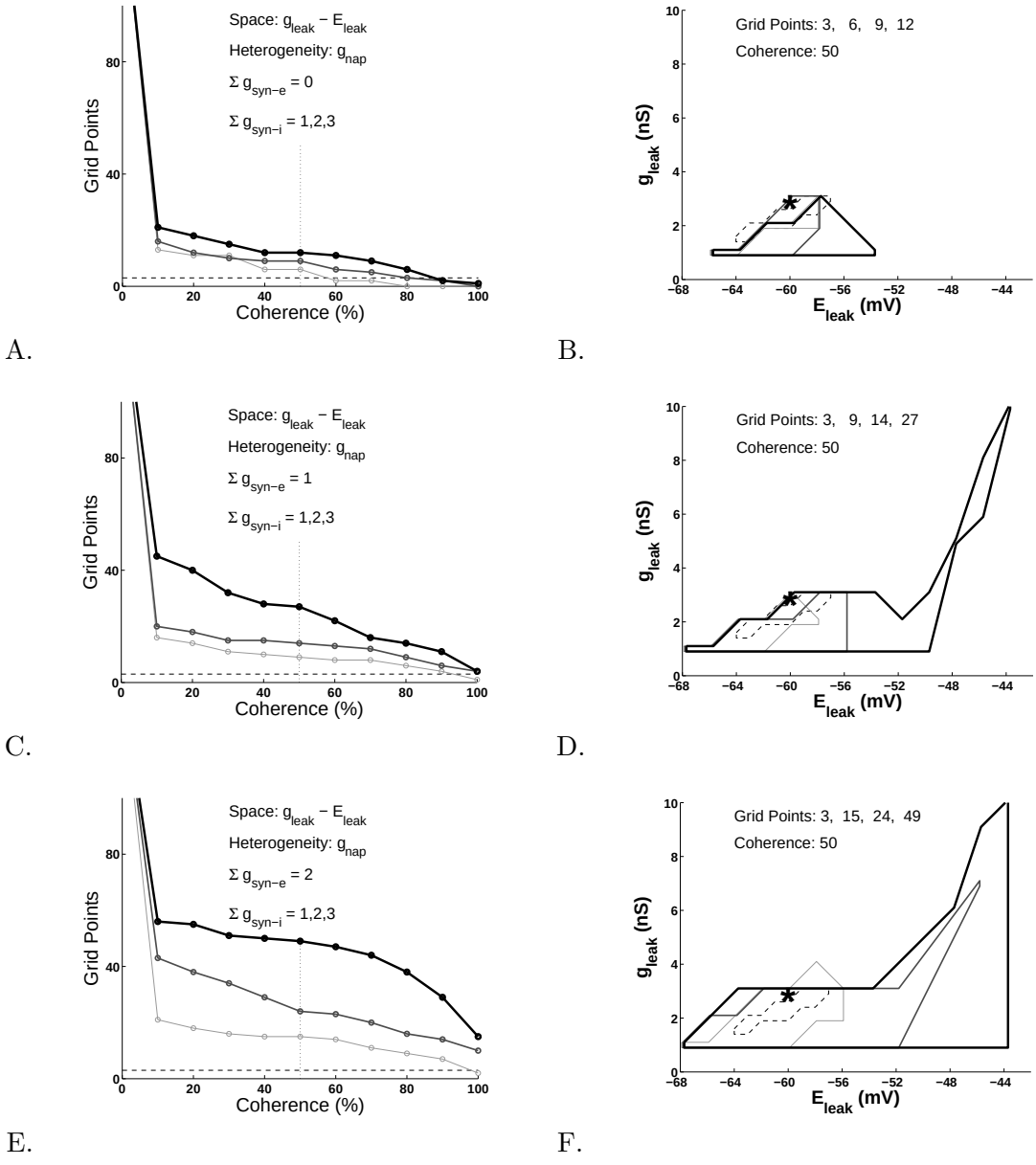


Figure 79: FPGA primary Test 2—$\bar{g}_{\mathrm{Leak}} - E_{\mathrm{Leak}}$ space for 40% intrinsic heterogeneity. Figures A, C, and E show the cumulative distribution of coherence measures for increasing values of ipsilateral excitation ($\Sigma\bar{g}_{\mathrm{syn-e}}$) and contralateral inhibition ($\Sigma\bar{g}_{\mathrm{syn-i}}$). The 0 percent coherence level represents every grid point (140). The horizontal dotted lines represent the case for the isolated (3). The vertical dashed lines indicate the 50 percent coherence level, and the corresponding burst-mapping diagrams are shown in figures B, D, and F.

Table 22: FPGA supplemental Test D—comparisons of the sizes of $\bar{g}_{\mathrm{NaP}} - E_{\mathrm{Leak}}$ bursting spaces for various levels of intrinsic heterogeneity for 50% coherence.

Intrinsic Heterogeneity	Test 1 0%	Test D 20%	Test 2 40%	Test D 60%
Figure	75	80	78	81
Grid Point–Coherence Slope	negligible	small	moderate	large
$\Sigma\bar{g}_{\mathrm{syn-e}} = 0$ nS, $\Sigma\bar{g}_{\mathrm{syn-i}} = 1, 2, 3$ nS	8,13,18	12,13,17	8,13,15	2,12,13
$\Sigma\bar{g}_{\mathrm{syn-e}} = 1$ nS, $\Sigma\bar{g}_{\mathrm{syn-i}} = 1, 2, 3$ nS	13,18,24	13,18,25	11,15,27	10,16,20
$\Sigma\bar{g}_{\mathrm{syn-e}} = 2$ nS, $\Sigma\bar{g}_{\mathrm{syn-i}} = 1, 2, 3$ nS	16,20,25	16,26,30	15,21,30	11,17,29

bursting region for small coherence values was smaller than that of an isolated neuron).

The results from these tests gave us the following principles: (1) The roles of the inhibitory and excitatory coupling were inconclusive. (2) To obtain 100% coherence, a homogenous network is required, but to obtain larger bursting regions, particularly for much smaller coherence values, a heterogeneous network is required. Too much heterogeneity, however, will eventually disrupt the network synchrony for any coherence value. (3) In a qualitative way, the steepness of the grid point–coherence slope is proportional to the parametric heterogeneity imposed in the network. Alternatively, for heterogeneous networks, the size of the bursting region is dependent on the coherence measure and the level of heterogeneity.

4.2.4 Primary Test 3—50% Connectivity

For our next study of heterogeneity, we were interested in testing the role of topological heterogeneity in a network of neurons by creating random sparse connectivity configurations. For our first set of tests, we implemented the following 50% connectivity scheme: For each neuron in the HCO, 50% (9/18) of its contralateral neurons were chosen to have double the synaptic weight (for the comparable test using the homogeneous configuration from primary test 1), and the other half of the synaptic connections were set to a weight of 0 nS. As a result of this normalization process that was previously described in the *Synaptic-Weight Model* part of Section 4.1, each neuron received the same total synaptic input as in the comparable homogeneous case. A pictorial of the *synaptic-weight model* was shown previously in Figure 66 for a variety of configurations. The most important point is that we

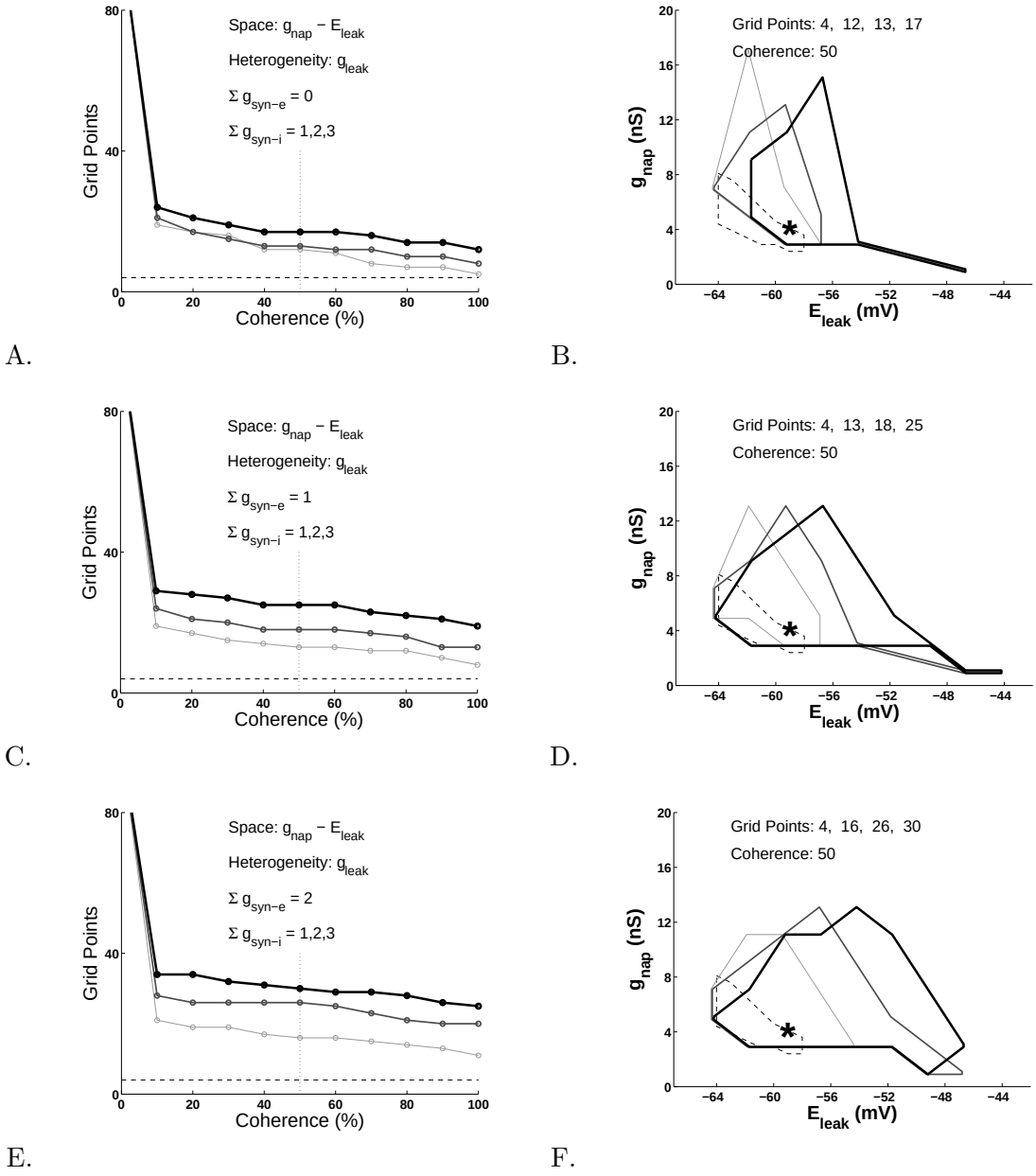


Figure 80: FPGA supplemental Test D—$\bar{g}_{\mathrm{NaP}} - E_{\mathrm{Leak}}$ space for 20% heterogeneity. Figures A, C, and E show the cumulative distribution of coherence measures for increasing values of ipsilateral excitation ($\Sigma\bar{g}_{\mathrm{syn-e}}$) and contralateral inhibition ($\Sigma\bar{g}_{\mathrm{syn-i}}$). The 0 percent coherence level represents every grid point (99). The horizontal dotted lines represent the case for the isolated neuron (4). The vertical dashed lines indicate the 50 percent coherence level, and the corresponding burst-mapping diagrams are shown in figures B, D, and F.

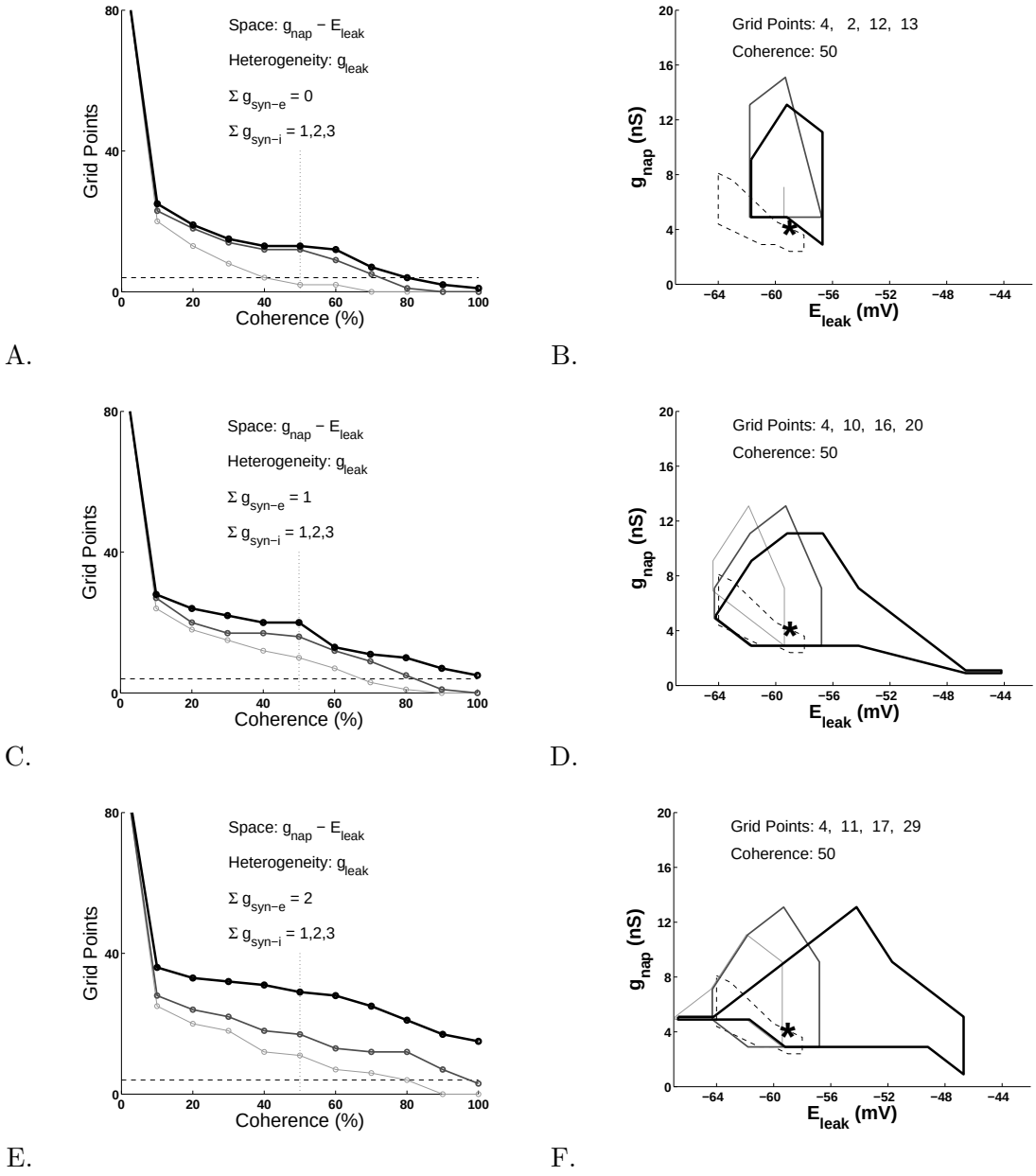


Figure 81: FPGA supplemental Test D—$\bar{g}_{\mathrm{NaP}} - E_{\mathrm{Leak}}$ space for 60% heterogeneity. Figures A, C, and E show the cumulative distribution of coherence measures for increasing values of ipsilateral excitation ($\Sigma\bar{g}_{\mathrm{syn-e}}$) and contralateral inhibition ($\Sigma\bar{g}_{\mathrm{syn-i}}$). The 0 percent coherence level represents every grid point (99). The horizontal dotted lines represent the case for the isolated neuron (4). The vertical dashed lines indicate the 50 percent coherence level, and the corresponding burst-mapping diagrams are shown in figures B, D, and F.

maintained an equivalent total synaptic input to every neuron in the HCO in every case, resulting in a conservation of synaptic weights. Because this process of randomly selecting half of the neurons from the other half of the HCO was done for each grid point in each of the burst-mapping diagrams, we were confident that we were not biasing our results with a single random topology and instead benefitted from averaging effects. In addition, since no parametric heterogeneity was imposed in the network, every neuron in this HCO was matched intrinsically.

The results of these tests are shown in Figures 82–84. As expected, the results are not significantly different from those obtained from the homogeneous tests (Figures 74–76): The grid point–coherence slope has a slight change from the homogeneous case, but the grid point values for 50% coherence are virtually the same, if not for the random noise in the system. This response was consistent with our belief that because all of the neurons were intrinsically matched, the inhibitory synaptic connections were just additive—the response to a distribution of synaptic values with some mean or to the same constant synaptic value for all connections will be virtually the same for both cases. Alternatively, if the neurons in a network were equivalent (intrinsically matched), then heterogeneity in the synaptic weights would not affect the network dynamics.

*Supplemental Test E—*4/18*,* 2/18*, and* 1/18 *Connectivity.* To make a more definitive conclusion regarding the theory that we formed from primary test 3, we performed similar tests for 4/18, 2/18, and 1/18 sparse connectivity probabilities in the $\bar{g}_{\mathrm{NaP}} - E_{\mathrm{Leak}}$ space for supplemental test E. The results for these tests are shown in Figures 85–87. As expected, we saw insignificant changes between these results and those of the 50% connectivity probability case (primary test 3) and the 100% connectivity probability case (primary test 1) because the response to a distribution of synaptic values with some mean or to the same constant synaptic value for all connections will be virtually the same for both cases. The grid point–coherence slopes are virtually zero and are nearly indistinguishable between all of the test cases. Table 23 shows the sizes of the $\bar{g}_{\mathrm{NaP}} - E_{\mathrm{Leak}}$ bursting regions for 50% coherence for all five topological configurations—18/18, 9/18, 4/18, 2/18, and 1/18. This table shows that the bursting regions are nearly identical for all levels of excitatory and inhibitory synaptic

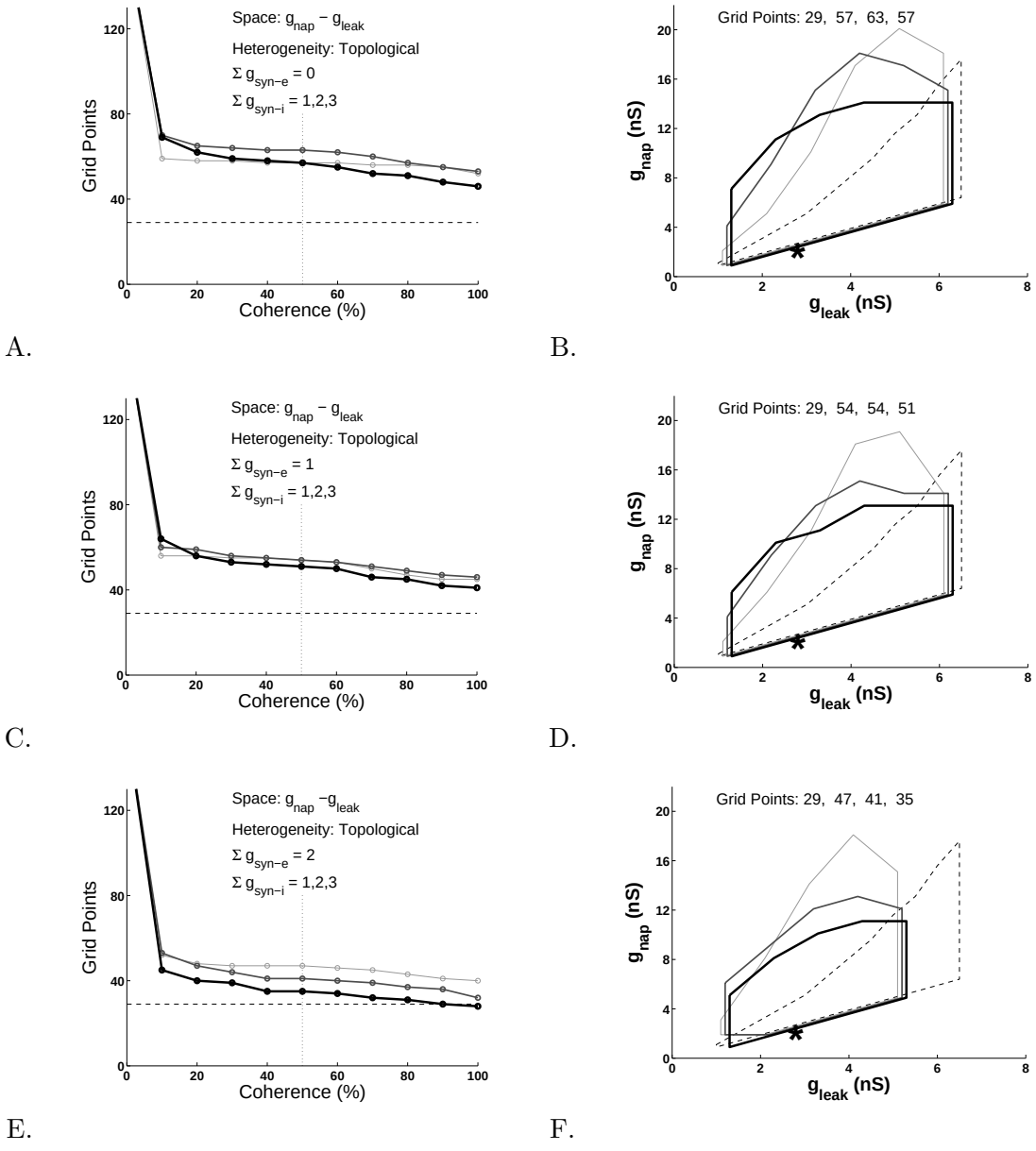


Figure 82: FPGA primary Test 3—$\bar{g}_{\mathrm{NaP}} - \bar{g}_{\mathrm{Leak}}$ space with 50% connectivity. Figures A, C, and E show the cumulative distribution of coherence measures for increasing values of ipsilateral excitation ($\Sigma\bar{g}_{\mathrm{syn-e}}$) and contralateral inhibition ($\Sigma\bar{g}_{\mathrm{syn-i}}$). The 0 percent coherence level represents every grid point (160). The horizontal dotted lines represent the case for the isolated neuron (29). The vertical dashed lines indicate the 50 percent coherence level, and the corresponding burst-mapping diagrams are shown in figures B, D, and F.

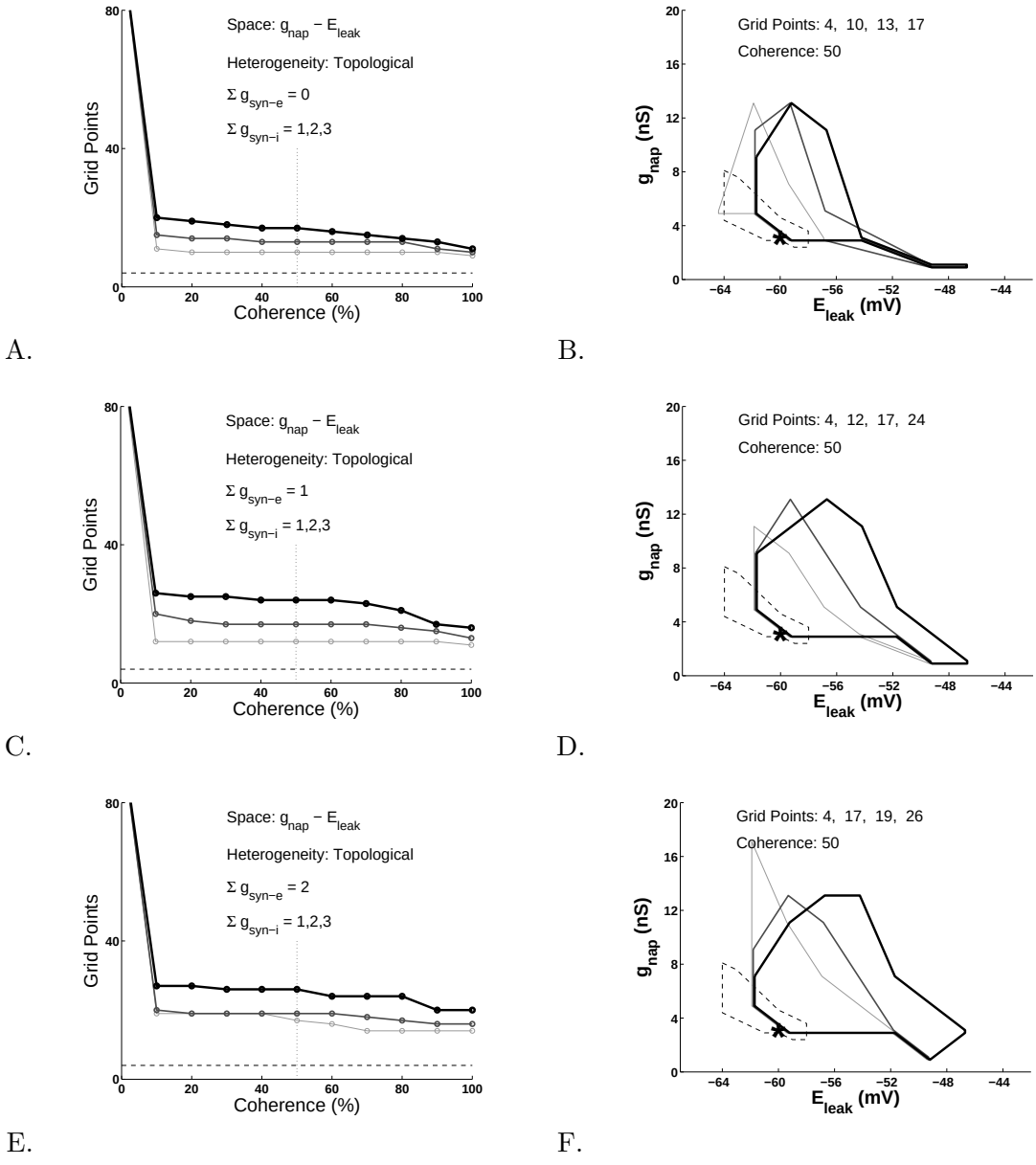


Figure 83: FPGA primary Test 3—$\bar{g}_{\mathrm{NaP}} - E_{\mathrm{Leak}}$ space with 50% connectivity. Figures A, C, and E show the cumulative distribution of coherence measures for increasing values of ipsilateral excitation ($\Sigma\bar{g}_{\mathrm{syn-e}}$) and contralateral inhibition ($\Sigma\bar{g}_{\mathrm{syn-i}}$). The 0 percent coherence level represents every grid point (99). The horizontal dotted lines represent the case for the isolated neuron (4). The vertical dashed lines indicate the 50 percent coherence level, and the corresponding burst-mapping diagrams are shown in figures B, D, and F.

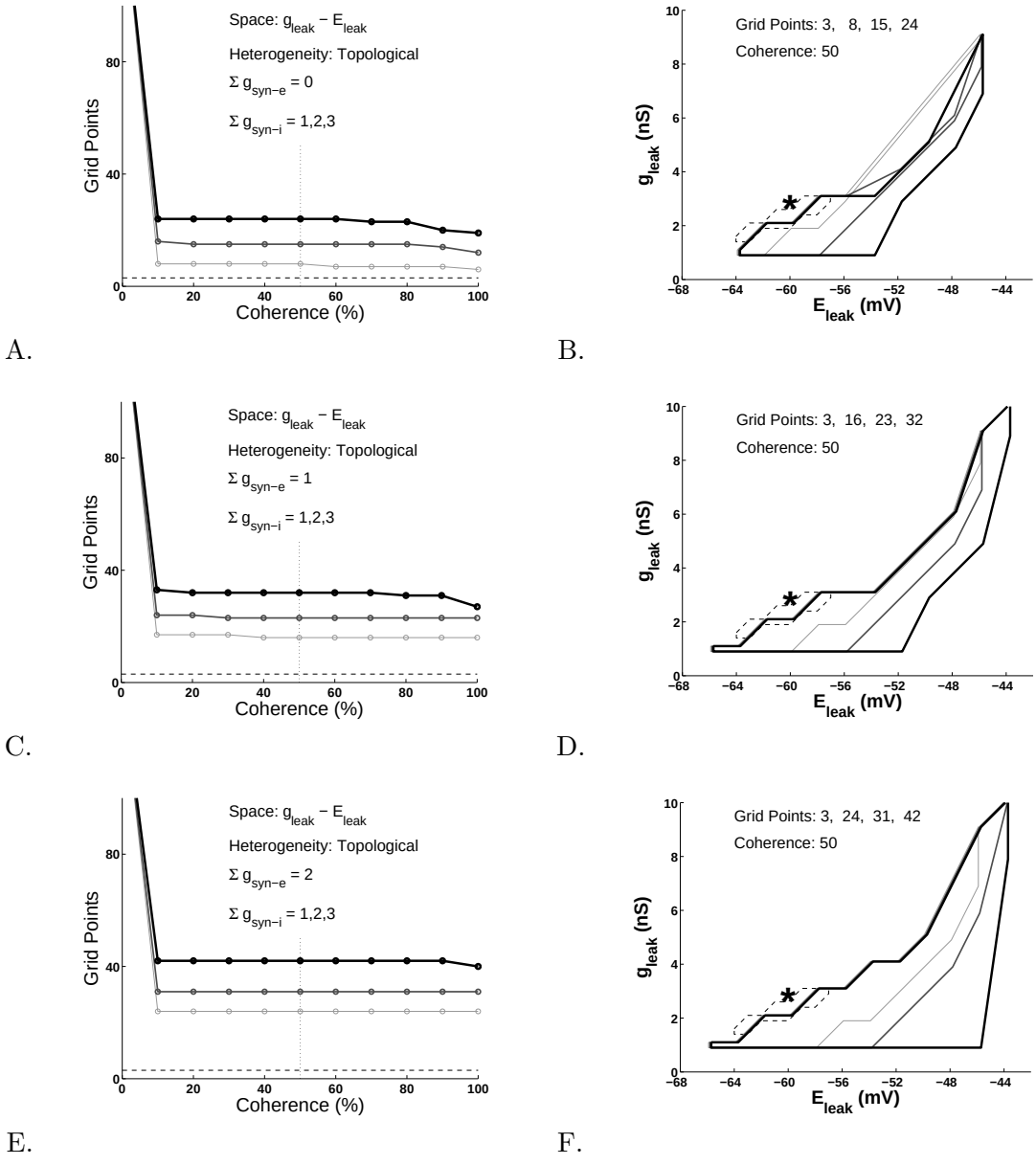


Figure 84: FPGA primary Test 3—$\bar{g}_{\mathrm{Leak}} - E_{\mathrm{Leak}}$ space with 50% connectivity. Figures A, C, and E show the cumulative distribution of coherence measures for increasing values of ipsilateral excitation ($\Sigma\bar{g}_{\mathrm{syn-e}}$) and contralateral inhibition ($\Sigma\bar{g}_{\mathrm{syn-i}}$). The 0 percent coherence level represents every grid point (140). The horizontal dotted lines represent the case for the isolated neuron (3). The vertical dashed lines indicate the 50 percent coherence level, and the corresponding burst-mapping diagrams are shown in figures B, D, and F.

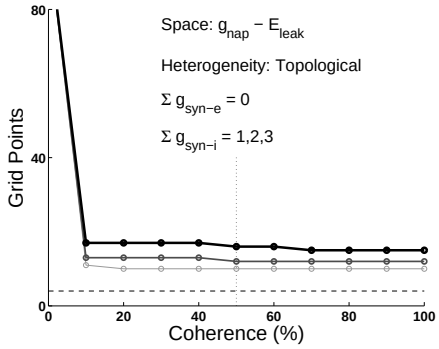
Table 23: FPGA supplemental Test E—comparisons of sizes of $\bar{g}_{\text{NaP}} - E_{\text{Leak}}$ bursting spaces for various connectivity configurations for 50% coherence.

Connectivity Probability	Test 1 18/18	Test 3 9/18	Test E 4/18	Test E 2/18	Test E 1/18
Figure	75	83	85	86	87
$\Sigma\bar{g}_{\text{syn-e}} = 0$ nS, $\Sigma\bar{g}_{\text{syn-i}} = 1, 2, 3$ nS	8,13,18	10,13,17	10,12,16	11,14,17	12,14,16
$\Sigma\bar{g}_{\text{syn-e}} = 1$ nS, $\Sigma\bar{g}_{\text{syn-i}} = 1, 2, 3$ nS	13,18,24	12,17,24	13,17,19	14,16,22	13,16,21
$\Sigma\bar{g}_{\text{syn-e}} = 2$ nS, $\Sigma\bar{g}_{\text{syn-i}} = 1, 2, 3$ nS	16,20,25	17,19,26	14,17,22	13,18,19	14,17,22

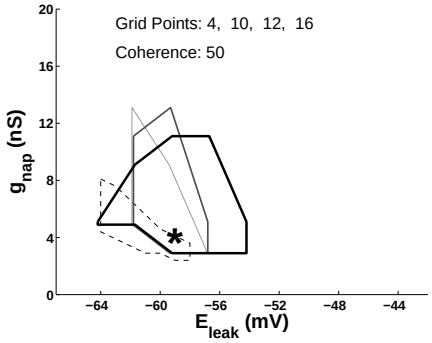
connections that were tested.

During our initial planning of the primary tests, we originally included one test to study how the role of 40% synaptic heterogeneity would affect the network. The relationship between this synaptic heterogeneity test and the other topological heterogeneity tests is shown pictorially in Figure 66. For this planned test, we would have used a 100% connectivity configuration, but each of the 18 contralateral inhibitory weights to each neuron would have required a normal distribution in which its coefficient of variation was 40%. As a result of the findings from the topological tests (primary test 3, supplemental test F), however, we removed the 40% synaptic heterogeneity test because of its redundancy, which would have resulted in another variation of topological heterogeneity as previously shown in Figure 66. The topological heterogeneity that we imposed on the synaptic network resulted in the following general claim: the response to a distribution of synaptic values with some mean or to the same constant synaptic value for all connections will be virtually the same for both cases.

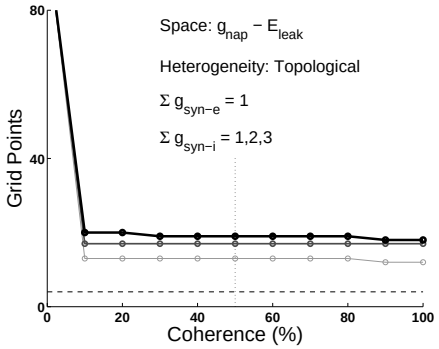
*Supplemental Test F—*9/18*,* 4/18*,* 2/18*, and* 1/18 *Connectivity with 40% Heterogeneity in* $\bar{g}_{\text{Leak}}$. After completing the intrinsic (primary test 2) and topological (primary test 3) tests, we studied the role of the combination of these two types of heterogeneity with supplemental test F. In general, the analysis of the simultaneous variation of multiple system parameters is more difficult than the sum of the analysis of the individual tests, but in this case, the analysis was simplified because the topological tests gave results similar to the homogeneous (control) tests. For this test, we applied sparse connection probabilities of 9/18, 4/18, 2/18, and 1/18 while simultaneously applying 40% heterogeneity to $\bar{g}_{\text{Leak}}$. The results for these tests are shown in Figures 88–91. We saw insignificant changes between



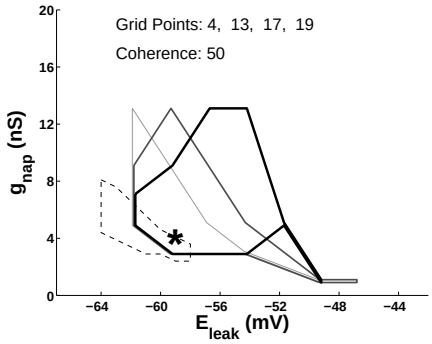
A.



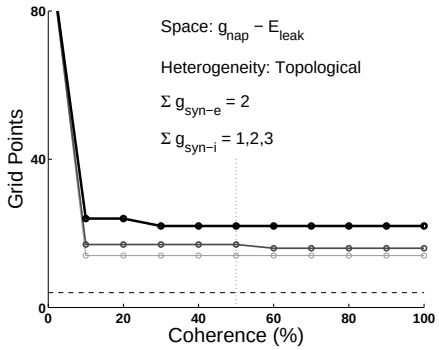
B.



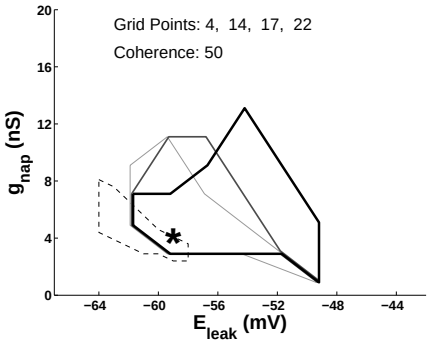
C.



D.

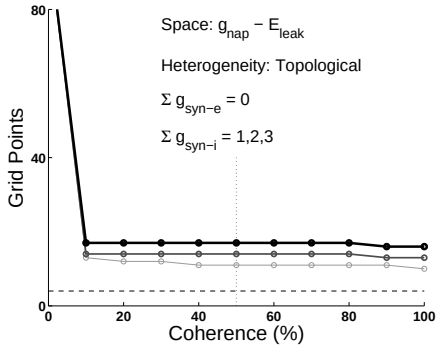


E.

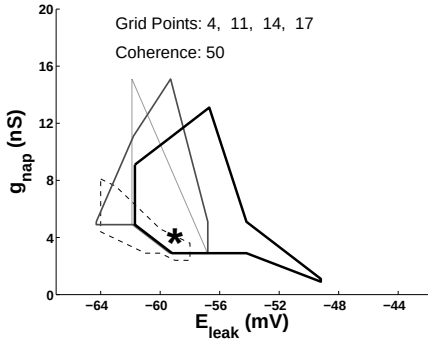


F.

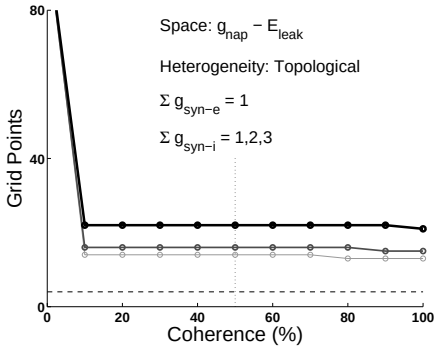
Figure 85: FPGA supplemental Test E—$\bar{g}_{\mathrm{NaP}} - E_{\mathrm{Leak}}$ space with 4/18 connectivity. Figures A, C, and E show the cumulative distribution of coherence measures for increasing values of ipsilateral excitation ($\Sigma\bar{g}_{\mathrm{syn-e}}$) and contralateral inhibition ($\Sigma\bar{g}_{\mathrm{syn-i}}$). The 0 percent coherence level represents every grid point (99). The horizontal dotted lines represent the case for the isolated neuron (4). The vertical dashed lines indicate the 50 percent coherence level, and the corresponding burst-mapping diagrams are shown in figures B, D, and F.



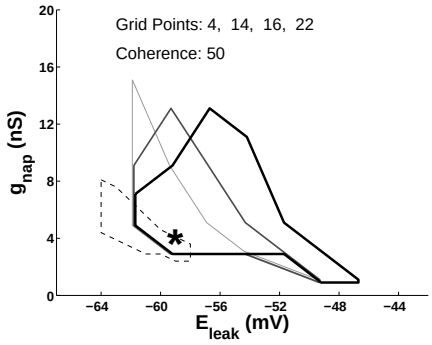
A.



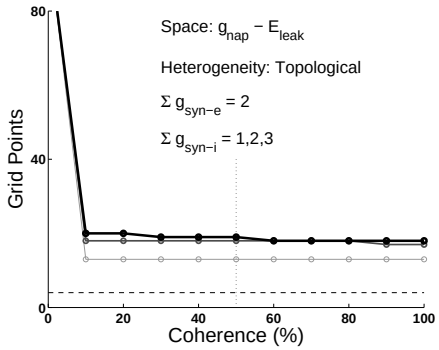
B.



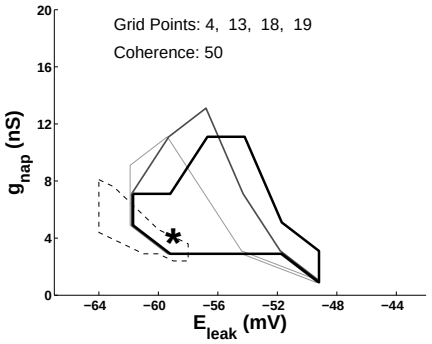
C.



D.

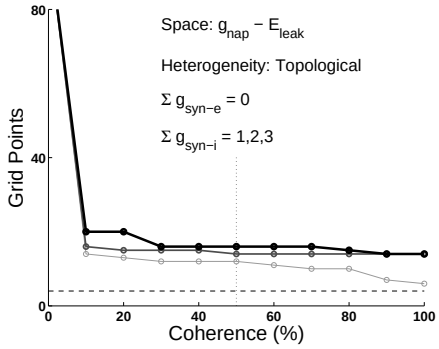


E.

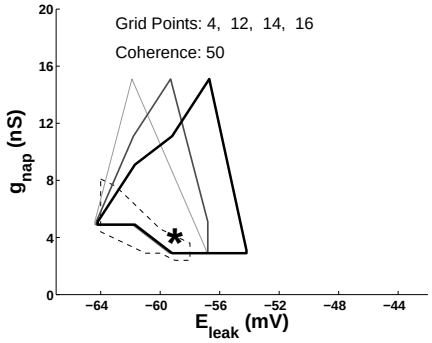


F.

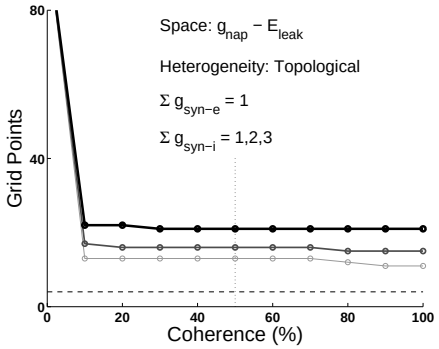
Figure 86: FPGA supplemental Test E—$\bar{g}_{\mathrm{NaP}} - E_{\mathrm{Leak}}$ space with 2/18 connectivity. Figures A, C, and E show the cumulative distribution of coherence measures for increasing values of ipsilateral excitation ($\Sigma\bar{g}_{\mathrm{syn-e}}$) and contralateral inhibition ($\Sigma\bar{g}_{\mathrm{syn-i}}$). The 0 percent coherence level represents every grid point (99). The horizontal dotted lines represent the case for the isolated neuron (4). The vertical dashed lines indicate the 50 percent coherence level, and the corresponding burst-mapping diagrams are shown in figures B, D, and F.



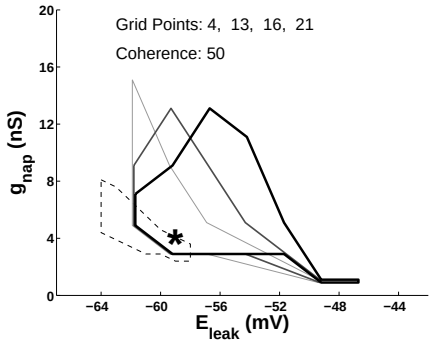
A.



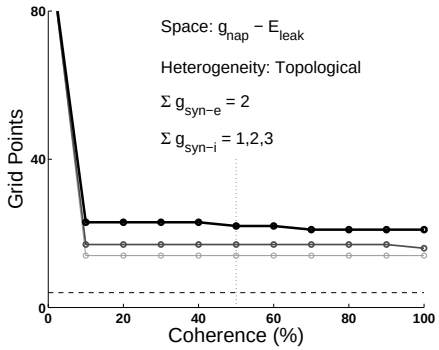
B.



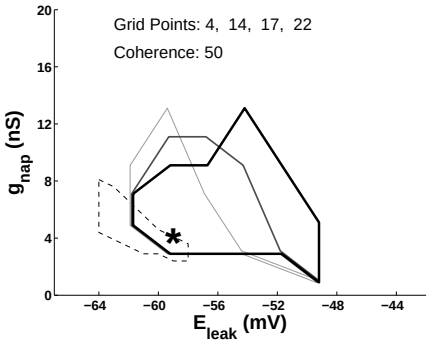
C.



D.



E.



F.

Figure 87: FPGA supplemental Test E—$\bar{g}_{\mathrm{NaP}} - E_{\mathrm{Leak}}$ space with 1/18 connectivity. Figures A, C, and E show the cumulative distribution of coherence measures for increasing values of ipsilateral excitation ($\Sigma\bar{g}_{\mathrm{syn-e}}$) and contralateral inhibition ($\Sigma\bar{g}_{\mathrm{syn-i}}$). The 0 percent coherence level represents every grid point (99). The horizontal dotted lines represent the case for the isolated neuron (4). The vertical dashed lines indicate the 50 percent coherence level, and the corresponding burst-mapping diagrams are shown in figures B, D, and F.

Table 24: FPGA supplemental Test F—comparisons of sizes of $\bar{g}_{\mathrm{NaP}} - E_{\mathrm{Leak}}$ bursting spaces for various connectivity probabilities and 40% heterogeneity in $\bar{g}_{\mathrm{Leak}}$. The three rows of data that are shown for each value of $\Sigma\bar{g}_{\mathrm{syn-e}}$ correspond to coherence values of 10% (top), 50%, and 90% (bottom).

Connectivity Probability	Test 2 18/18	Test F 9/18	Test F 4/18	Test F 2/18	Test F 1/18
Figure	78	88	89	90	91
$\Sigma\bar{g}_{\mathrm{syn-e}} = 0$ nS, $\Sigma\bar{g}_{\mathrm{syn-i}} = 1, 2, 3$ nS	18,25,21 8,13,15 0,3,8	19,26,25 8,12,15 0,4,9	22,26,24 7,12,15 0,4,8	24,27,23 9,8,11 0,2,7	23,19,21 9,9,12 0,1,1
$\Sigma\bar{g}_{\mathrm{syn-e}} = 1$ nS, $\Sigma\bar{g}_{\mathrm{syn-i}} = 1, 2, 3$ nS	23,25,29 11,15,27 6,7,13	25,26,29 13,16,24 5,9,14	23,23,30 15,14,24 5,6,12	23,23,24 14,14,23 6,8,9	21,22,28 13,17,19 3,7,9
$\Sigma\bar{g}_{\mathrm{syn-e}} = 2$ nS, $\Sigma\bar{g}_{\mathrm{syn-i}} = 1, 2, 3$ nS	28,29,36 15,21,30 8,12,22	29,26,35 16,21,29 7,12,21	23,27,32 14,23,29 8,13,21	26,27,34 18,22,31 8,14,23	25,31,38 17,22,32 6,7,20

these results and those of primary test 2, which for this supplemental test is the control data. This was expected because we previously showed that the application of topological heterogeneity, in which the synaptic input to any neuron was conserved, has little effect on the population-level bursting measure, resulting in similar grid–point coherence slopes for all cases. Table 24 shows the sizes of the $\bar{g}_{\mathrm{NaP}} - E_{\mathrm{Leak}}$ bursting regions for 40% heterogeneity for all five topological configurations—18/18, 9/18, 4/18, 2/18, and 1/18. This table shows that the bursting regions are nearly identical for all levels of excitatory and inhibitory synaptic connections. Because of the parametric heterogeneity, the sizes of the bursting regions depend on the coherence values, and therefore, the table includes sizes for coherence values of 10%, 50%, and 90% so that broad comparisons can be made.

4.2.5 Primary Test 4—Non-Canonical Point with Parametric Heterogeneity

For our next primary study of heterogeneity and our first one at a non-canonical point, we were interested in determining the role of the intrinsic point in parameter space—that is, whether the directions of the changes to the sizes of the bursting regions for different synaptic-weight configurations were dependent on the canonical point. To do so, we analyzed the role that 40% heterogeneity imposed on the intrinsic parameters had on a different point in parameter space. The following were the published canonical points [8]:

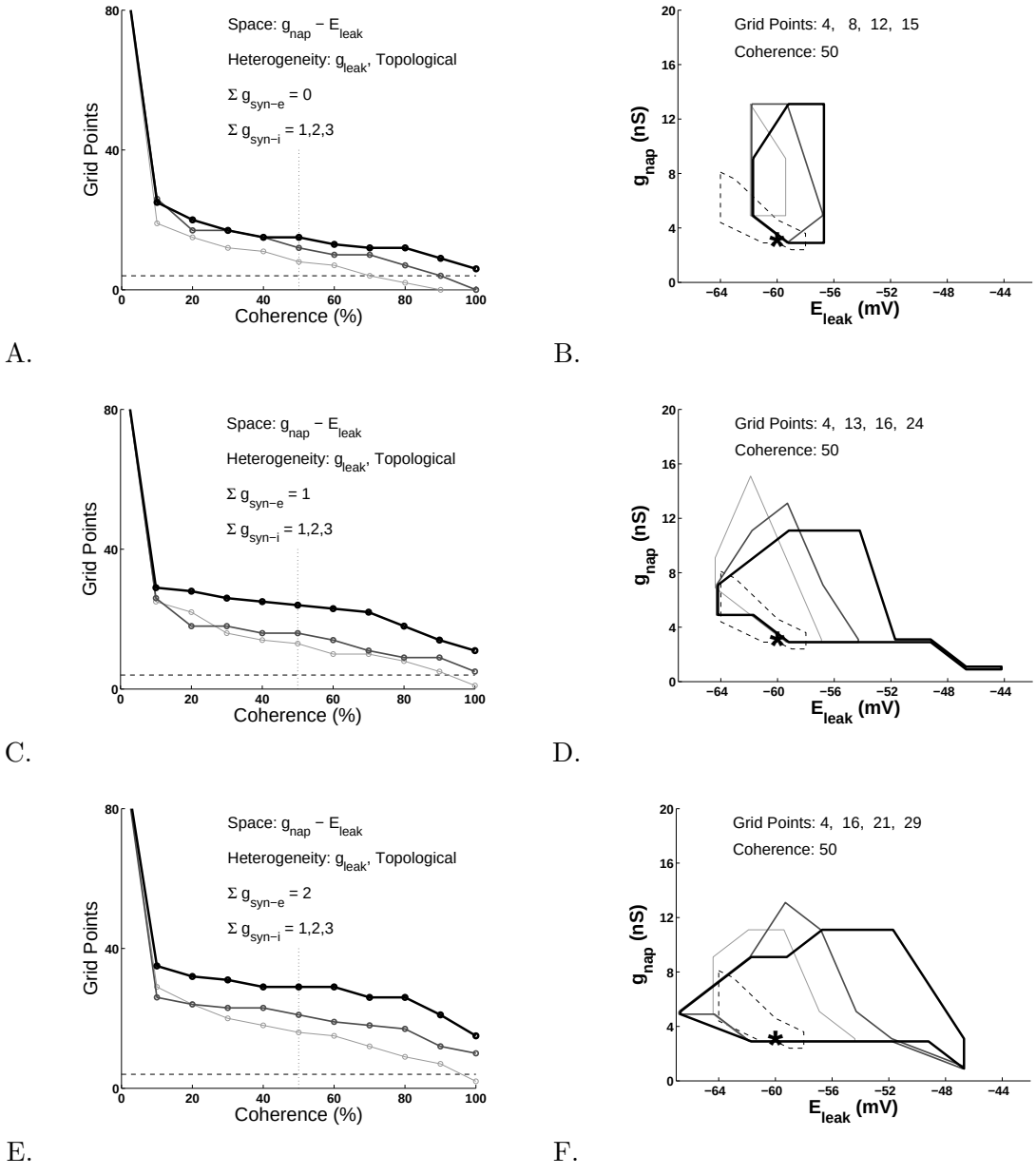


Figure 88: FPGA supplemental Test F—$\bar{g}_{\mathrm{NaP}} - E_{\mathrm{Leak}}$ space with 9/18 connectivity and 40% heterogeneity in $\bar{g}_{\mathrm{Leak}}$. Figures A, C, and E show the cumulative distribution of coherence measures for increasing values of ipsilateral excitation ($\Sigma\bar{g}_{\mathrm{syn-e}}$) and contralateral inhibition ($\Sigma\bar{g}_{\mathrm{syn-i}}$). The 0 percent coherence level represents every grid point (99). The horizontal dotted lines represent the case for the isolated neuron (4). The vertical dashed lines indicate the 50 percent coherence level, and the corresponding burst-mapping diagrams are shown in figures B, D, and F.

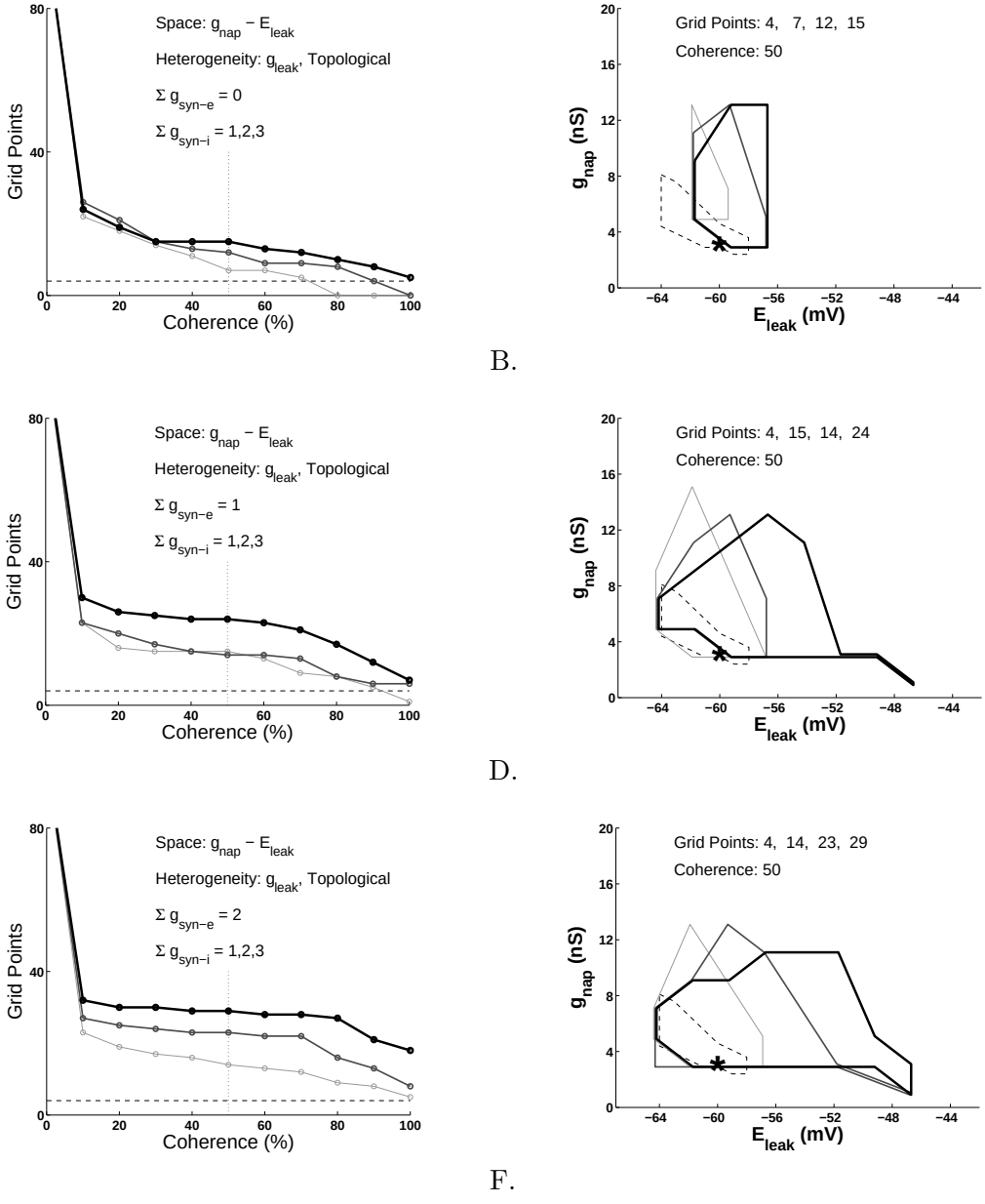


Figure 89: FPGA supplemental Test F—$\bar{g}_{\mathrm{NaP}} - E_{\mathrm{Leak}}$ space with 4/18 connectivity and 40% heterogeneity in $\bar{g}_{\mathrm{Leak}}$. Figures A, C, and E show the cumulative distribution of coherence measures for increasing values of ipsilateral excitation ($\Sigma\bar{g}_{\mathrm{syn-e}}$) and contralateral inhibition ($\Sigma\bar{g}_{\mathrm{syn-i}}$). The 0 percent coherence level represents every grid point (99). The horizontal dotted lines represent the case for the isolated neuron (4). The vertical dashed lines indicate the 50 percent coherence level, and the corresponding burst-mapping diagrams are shown in figures B, D, and F.

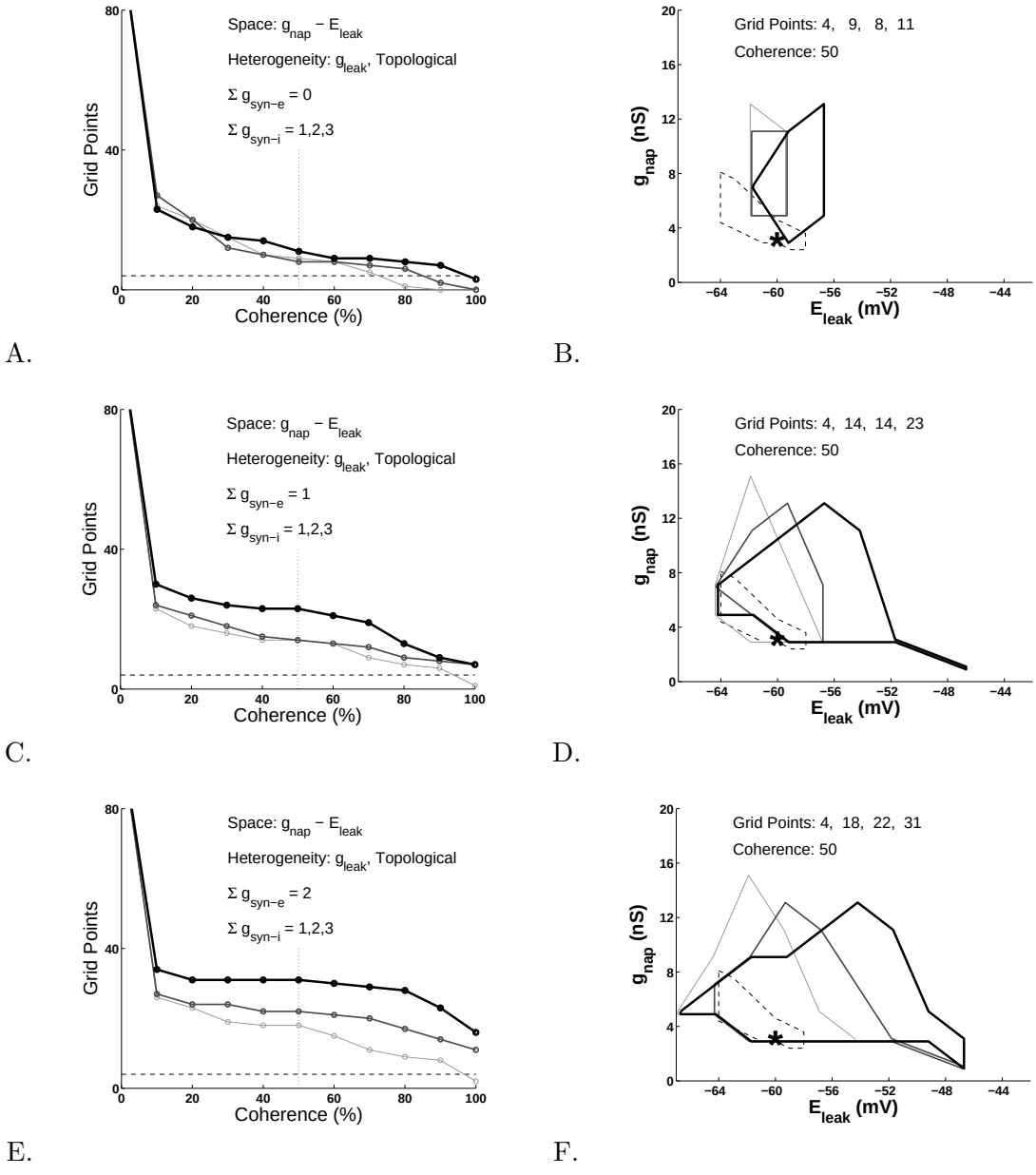


Figure 90: FPGA supplemental Test F—$\bar{g}_{\mathrm{NaP}} - E_{\mathrm{Leak}}$ space with 2/18 connectivity and 40% heterogeneity in $\bar{g}_{\mathrm{Leak}}$. Figures A, C, and E show the cumulative distribution of coherence measures for increasing values of ipsilateral excitation ($\Sigma\bar{g}_{\mathrm{syn-e}}$) and contralateral inhibition ($\Sigma\bar{g}_{\mathrm{syn-i}}$). The 0 percent coherence level represents every grid point (99). The horizontal dotted lines represent the case for the isolated neuron (4). The vertical dashed lines indicate the 50 percent coherence level, and the corresponding burst-mapping diagrams are shown in figures B, D, and F.

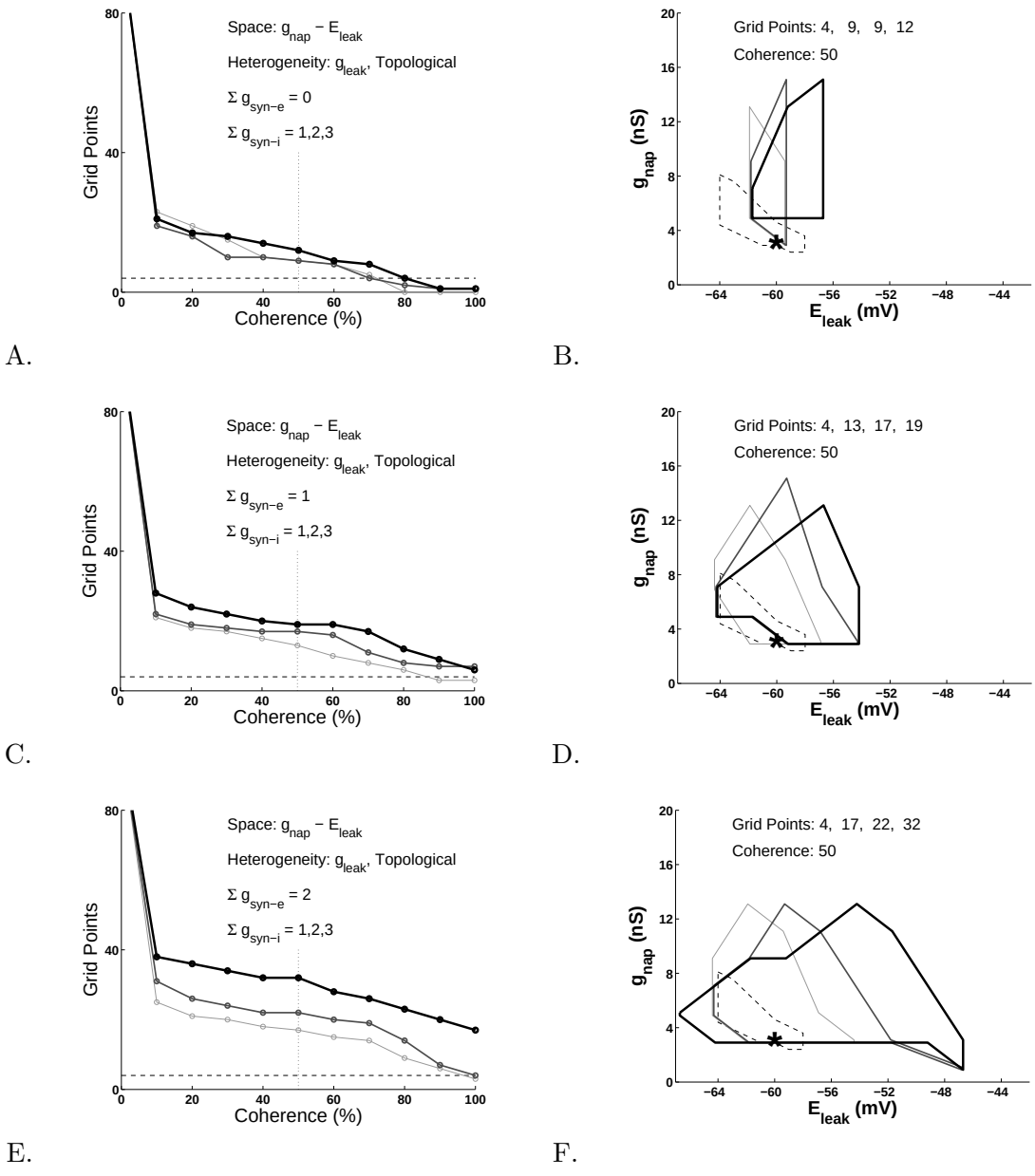


Figure 91: FPGA supplemental Test F—$\bar{g}_{\mathrm{NaP}} - E_{\mathrm{Leak}}$ space with 1/18 connectivity and 40% heterogeneity in $\bar{g}_{\mathrm{Leak}}$. Figures A, C, and E show the cumulative distribution of coherence measures for increasing values of ipsilateral excitation ($\Sigma\bar{g}_{\mathrm{syn-e}}$) and contralateral inhibition ($\Sigma\bar{g}_{\mathrm{syn-i}}$). The 0 percent coherence level represents every grid point (99). The horizontal dotted lines represent the case for the isolated neuron (4). The vertical dashed lines indicate the 50 percent coherence level, and the corresponding burst-mapping diagrams are shown in figures B, D, and F.

$\bar{g}_{\text{NaP}} = 2.8$ nS, $\bar{g}_{\text{Leak}} = 2.8$ nS, and $E_{\text{Leak}} = -60$ mV. As was shown in Figure 72 from preliminary test B (the analysis of the isolated neuron), the largest bursting regions do not occur with the canonical points, which are on the edges of the bursting regions; because of the quantization of the grid points, the canonical points are actually shown to be outside of the bursting regions. Based on the results from the previous tests, we decided to use the following "non-canonical" parameter values for this primary test 4: $\bar{g}_{\text{NaP}} = 4.0$ nS, $\bar{g}_{\text{Leak}} = 2.0$ nS, and $E_{\text{Leak}} = -59$ mV. The results were mixed (Figures 92–94) and should be compared to the heterogeneity results from primary test 2 shown in Figures 77–79: (1) In the $\bar{g}_{\text{NaP}} - \bar{g}_{\text{Leak}}$ space, the bursting regions are significantly larger for the new value of E_{Leak}, and in fact, the difference would have been even larger except the pre-defined grid was not large enough to capture the entire bursting space. Also, the characteristic steep grid point–coherence slope is evident for both canonical points. (2) In the $\bar{g}_{\text{NaP}} - E_{\text{Leak}}$ space, the results are virtually identical for both values of $\bar{g}_{\text{Leak}}$, and in the $\bar{g}_{\text{Leak}} - E_{\text{Leak}}$ space, the bursting regions are moderately larger for the new, increased value of $\bar{g}_{\text{NaP}}$. In each of these spaces, a sensitivity analysis would show that the changes to the sizes of the bursting regions for different synaptic-weight configurations are all in the same direction for both values of the intrinsic parameter.

These results were important because they showed that the canonical parameter set did not result in the largest bursting regions; because of this *single* example, we were not required to examine the vast parameter space any farther. In fact, using our stochastic gradient-descent algorithm (Section 3.1), we previously showed that the canonical point was arbitrary from a modeling perspective and that an arbitrarily large number of points in parameter space would result in a time trace of the neural output, V_{mem}, that was similar to the one at the canonical point. Finding the point in parameter space, however, that results in the largest two-dimensional bursting space is way beyond the scope of our work and the capabilities of our testing resources. These results also demonstrated another important result regarding our conclusions from this work: although we used three different two-dimensional bursting regions to reduce bias with our results, we were still biased at the canonical point in parameter space, resulting in location-specific, not necessarily general,

findings. For example, in the $\bar{g}_{\mathrm{NaP}} - \bar{g}_{\mathrm{Leak}}$ space for $\Sigma\bar{g}_{\mathrm{syn-e}} = 2$ and $\Sigma\bar{g}_{\mathrm{syn-i}} = 1 - 2$, the number of grid points *decreased* from 61 to 56 at the canonical point and *increased* from 71 to 78 at the non-canonical point, showing inconclusive results with the inhibitory synaptic connections. Although we could not find any clear examples from our data in which a change in $\Sigma\bar{g}_{\mathrm{syn-e}}$ increased or decreased the size of the bursting region at the canonical point (primary test 2) but gave the opposite result at the non-canonical point (primary test 4), we were confident that this result with the excitatory synaptic connections could still be obtained in the vast parameter space of this model.

4.2.6 Primary Test 5—Non-Canonical Point with Parametric and Topological Heterogeneity

For our final primary study of heterogeneity and our second at the non-canonical point in parameter space, we wanted to try to generalize some of the results found at the canonical point. Specifically, we wanted to simultaneously test the role of parametric and topological heterogeneity at a non-canonical point in parameter space. In general, the analysis of the simultaneous variation of multiple system parameters is more difficult than the sum of the analysis of the individual tests, but in this case, the analysis was simplified because the topological tests gave results similar to the homogeneous (control) tests. To perform this test, we analyzed the role that 40% heterogeneity imposed on the intrinsic parameters combined with randomly disconnecting half of the contralateral synaptic connections to a particular neuron at a different, non-canonical point in parameter space. The following were the published canonical points [8]: $\bar{g}_{\mathrm{NaP}} = 2.8$ nS, $\bar{g}_{\mathrm{Leak}} = 2.8$ nS, and $E_{\mathrm{Leak}} = -60$ mV. We used the same non-canonical point from primary test 4: $\bar{g}_{\mathrm{NaP}} = 4.0$ nS, $\bar{g}_{\mathrm{Leak}} = 2.0$ nS, and $E_{\mathrm{Leak}} = -59$ mV. As expected, the results shown in Figures 95–97 are similar to those from primary test 4 (Figures 92–94), the control data for this test. We expected these results because we previously found from primary test 3 that imposing topological heterogeneity in a way that maintains a constant synaptic input to each neuron will largely not affect the population-based bursting measure. Table 25 shows the sizes of the $\bar{g}_{\mathrm{NaP}} - E_{\mathrm{Leak}}$ bursting regions for 40% heterogeneity for a synaptic probability of 18/18 (primary test 4) and 9/18 (primary test 5). This table shows that the bursting regions are nearly identical for all

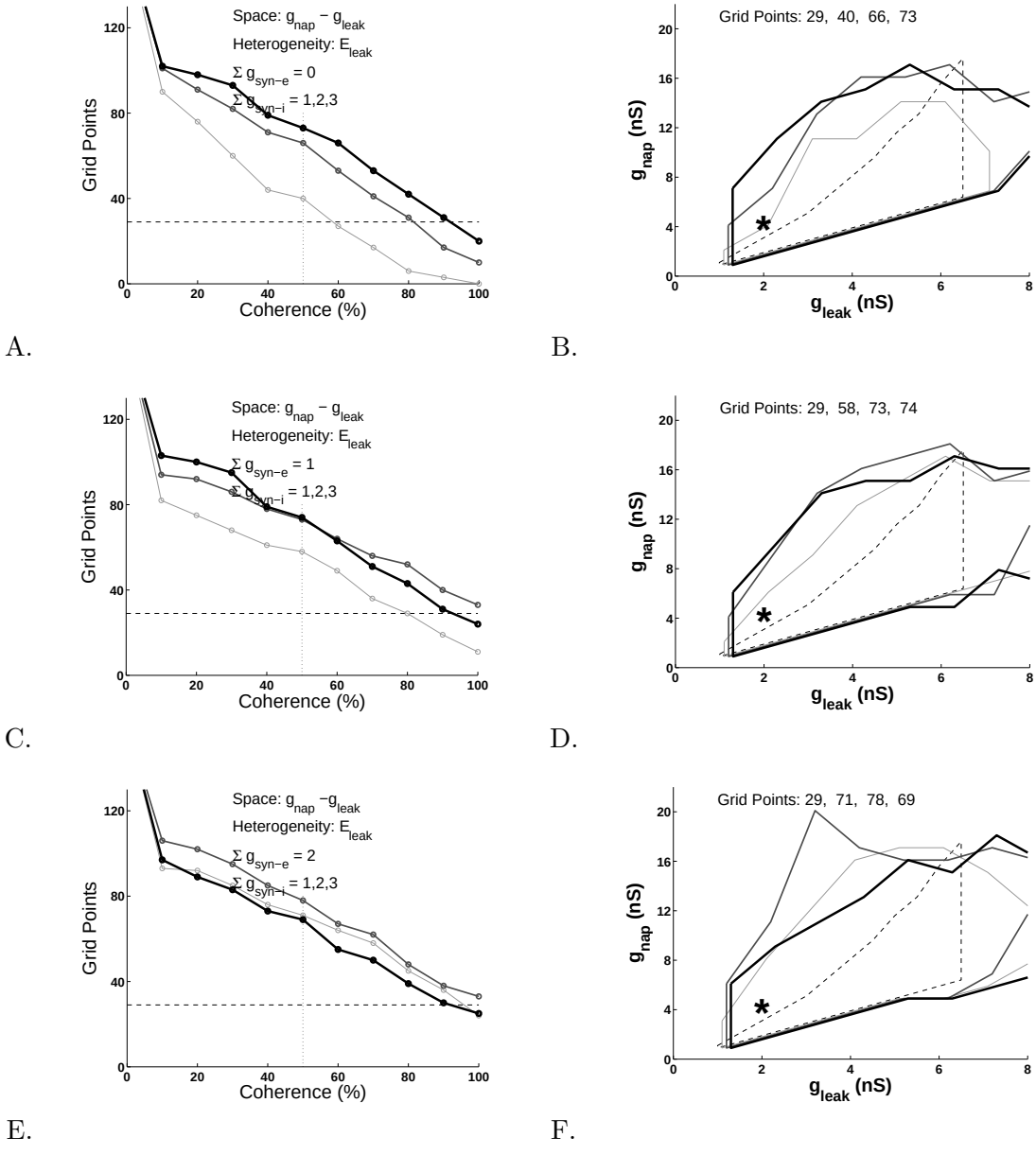
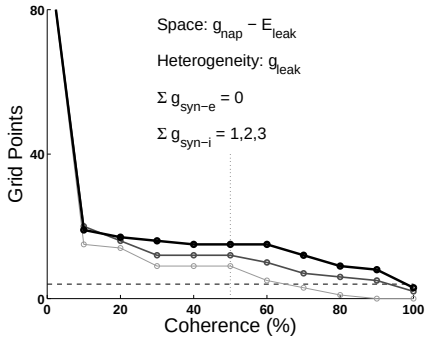
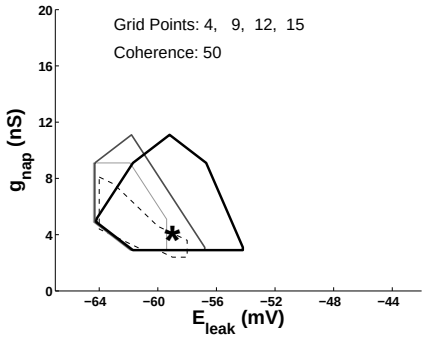


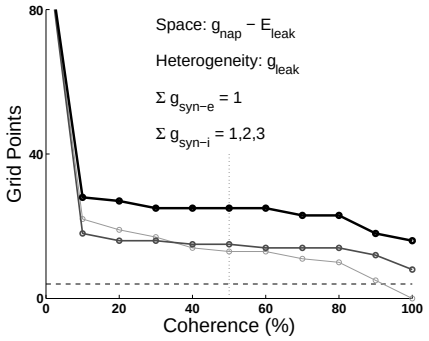
Figure 92: FPGA primary Test 4—$\bar{g}_{\mathrm{NaP}} - \bar{g}_{\mathrm{Leak}}$ space using a non-canonical point with parametric heterogeneity. Figures A, C, and E show the cumulative distribution of coherence measures for increasing values of ipsilateral excitation ($\Sigma\bar{g}_{\mathrm{syn-e}}$) and contralateral inhibition ($\Sigma\bar{g}_{\mathrm{syn-i}}$). The 0 percent coherence level represents every grid point (160). The horizontal dotted lines represent the case for the isolated neuron (29). The vertical dashed lines indicate the 50 percent coherence level, and the corresponding burst-mapping diagrams are shown in figures B, D, and F.



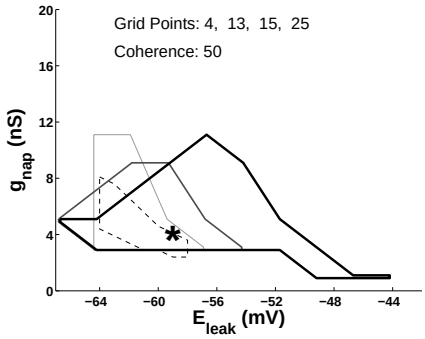
A.



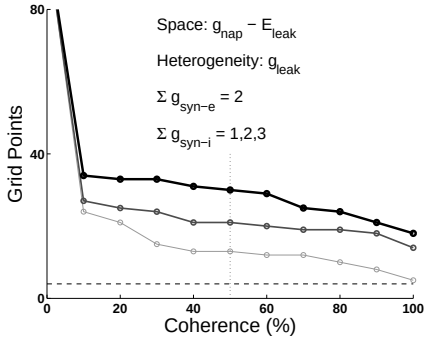
B.



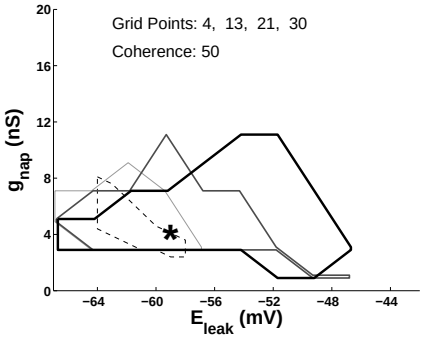
C.



D.

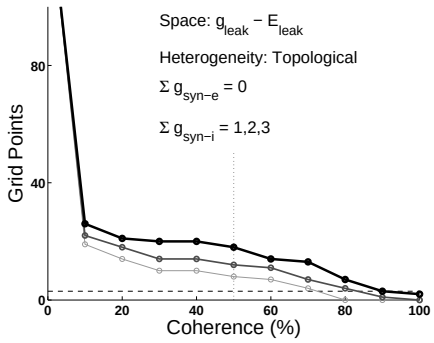


E.

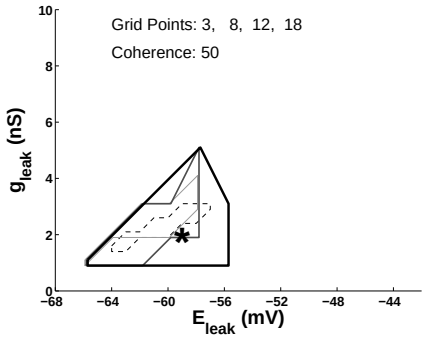


F.

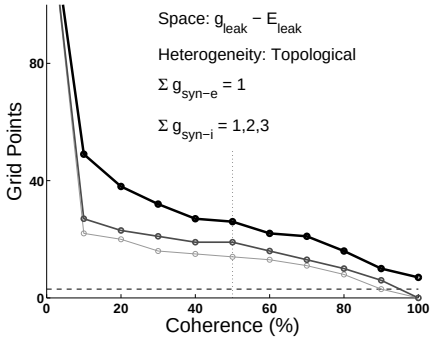
Figure 93: FPGA primary Test 4—$\bar{g}_{\mathrm{NaP}} - E_{\mathrm{Leak}}$ space using a non-canonical point with parametric heterogeneity. Figures A, C, and E show the cumulative distribution of coherence measures for increasing values of ipsilateral excitation ($\Sigma\bar{g}_{\mathrm{syn-e}}$) and contralateral inhibition ($\Sigma\bar{g}_{\mathrm{syn-i}}$). The 0 percent coherence level represents every grid point (99). The horizontal dotted lines represent the case for the isolated neuron (4). The vertical dashed lines indicate the 50 percent coherence level, and the corresponding burst-mapping diagrams are shown in figures B, D, and F.



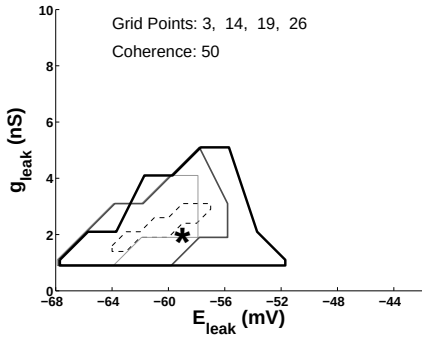
A.



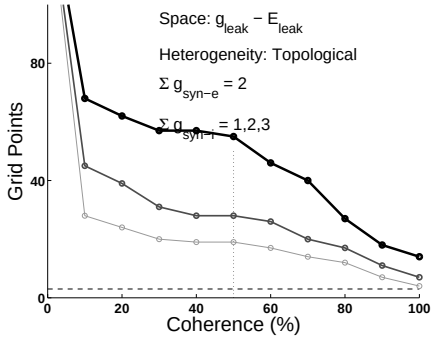
B.



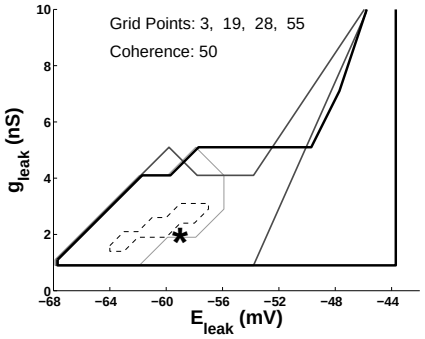
C.



D.



E.



F.

Figure 94: FPGA primary Test 4—$\bar{g}_{\mathrm{Leak}} - E_{\mathrm{Leak}}$ space using a non-canonical point with parametric heterogeneity. Figures A, C, and E show the cumulative distribution of coherence measures for increasing values of ipsilateral excitation ($\Sigma\bar{g}_{\mathrm{syn-e}}$) and contralateral inhibition ($\Sigma\bar{g}_{\mathrm{syn-i}}$). The 0 percent coherence level represents every grid point (140). The horizontal dotted lines represent the case for the isolated neuron (3). The vertical dashed lines indicate the 50 percent coherence level, and the corresponding burst-mapping diagrams are shown in figures B, D, and F.

Table 25: FPGA primary Test 5—comparisons of the sizes of the $\bar{g}_{\mathrm{NaP}} - E_{\mathrm{Leak}}$ bursting spaces using a non-canonical point and 40% heterogeneity in $\bar{g}_{\mathrm{Leak}}$ for 18/18 and 9/18 connectivity probabilities. The three rows of data that are shown for each value of $\Sigma\bar{g}_{\mathrm{syn-e}}$ correspond to coherence values of 10% (top), 50%, and 90% (bottom).

Connectivity Probability	Test 4 18/18	Test 5 9/18
Figure	93	96
$\Sigma\bar{g}_{\mathrm{syn-e}} = 0$ nS, $\Sigma\bar{g}_{\mathrm{syn-i}} = 1, 2, 3$ nS	15,20,19 9,12,15 0,5,8	18,25,26 8,12,15 0,4,10
$\Sigma\bar{g}_{\mathrm{syn-e}} = 1$ nS, $\Sigma\bar{g}_{\mathrm{syn-i}} = 1, 2, 3$ nS	22,18,28 13,15,25 5,12,18	22,23,29 12,14,24 5,10,13
$\Sigma\bar{g}_{\mathrm{syn-e}} = 2$ nS, $\Sigma\bar{g}_{\mathrm{syn-i}} = 1, 2, 3$ nS	24,27,34 13,21,30 8,18,21	22,28,38 14,22,29 8,15,20

levels of excitatory and inhibitory synaptic connections that were tested. Because of the parametric heterogeneity, the sizes of the bursting regions depend on the coherence values, and therefore, the table includes sizes for coherence values of 10%, 50%, and 90% so that a broad comparison can be made.

4.3 Summary

We studied the role of a variety of forms of heterogeneity in a network of neurons. After obtaining our control results from the homogeneous configuration, we move our test point in three different directions using the canonical point in parameter space. (1) For the homogeneous configuration, all of the intrinsic parameters were identical across all neurons. For our first heterogeneity study, we replaced identical parameter sets with normal distributions and studied their effects for a variety of standard deviations, in which the zero-standard deviation case was the homogeneous configuration. (2) For the homogeneous configuration, the synaptic network was fully-connected. For our next heterogeneity study, we started to randomly remove connections (and increased the weights of the remaining non-zero connections) and studied their effects for a variety of configurations. (3) For the homogeneous configuration, the published intrinsic values were used as the canonical points. For our final heterogeneity study, we changed those points slightly and obtained results for a new point

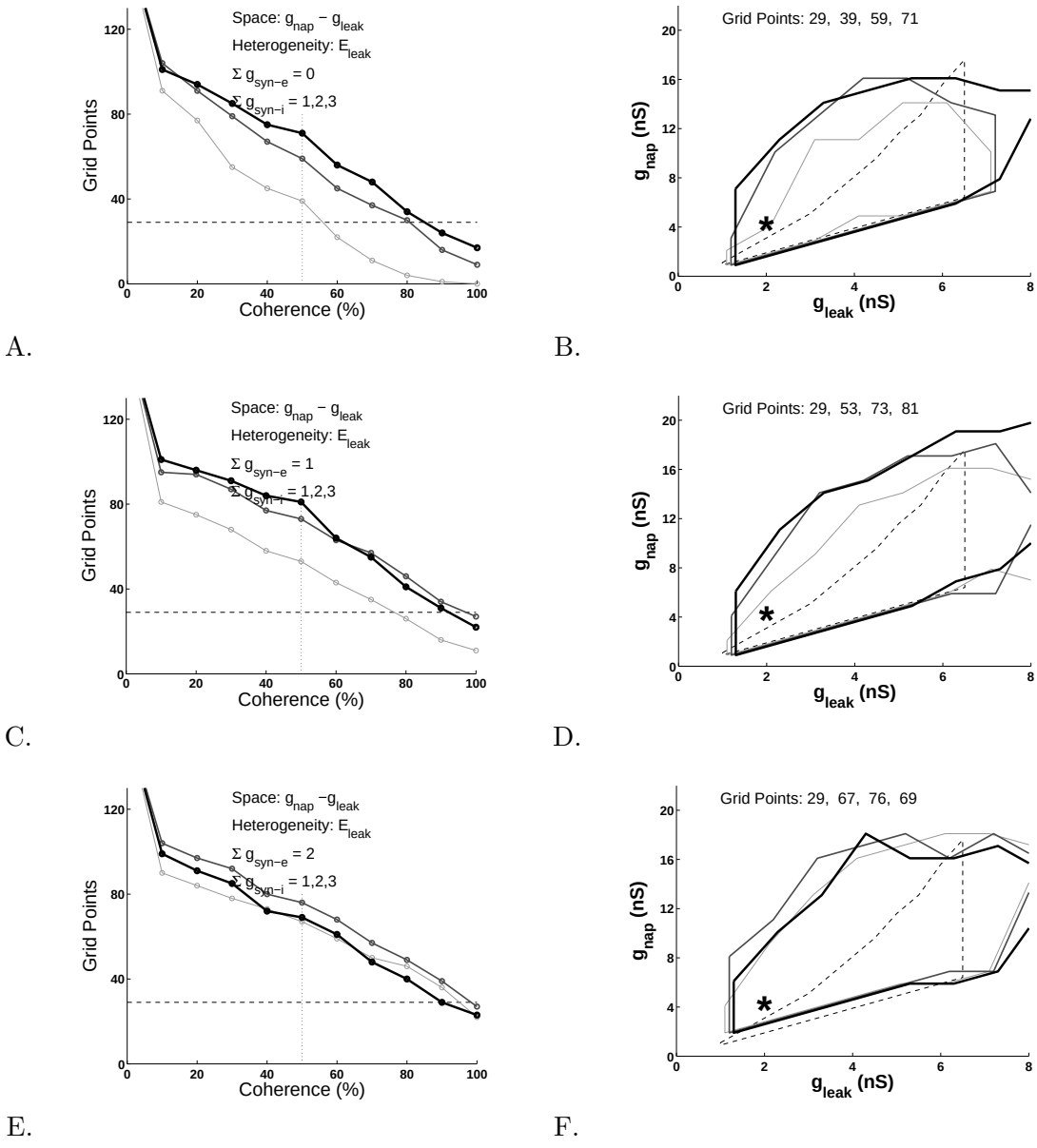
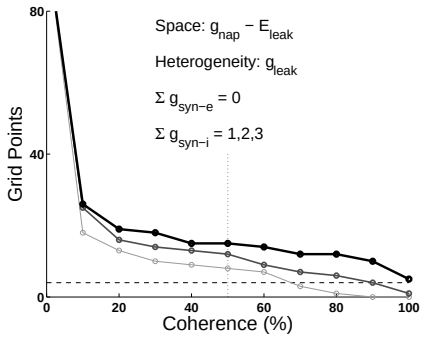
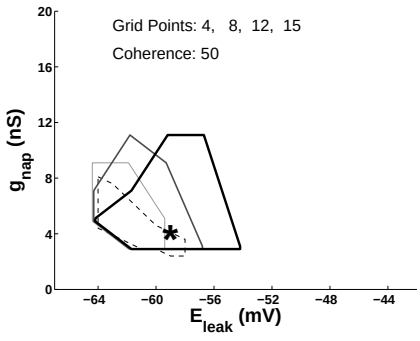


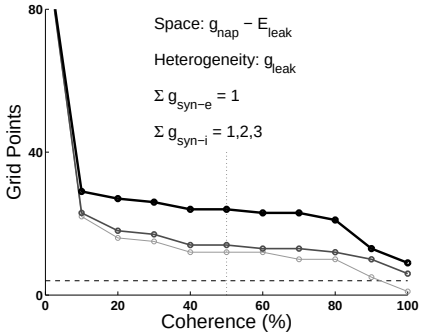
Figure 95: FPGA primary Test 5—$\bar{g}_{\mathrm{NaP}} - \bar{g}_{\mathrm{Leak}}$ space using a non-canonical point with parametric and topological heterogeneity. Figures A, C, and E show the cumulative distribution of coherence measures for increasing values of ipsilateral excitation ($\Sigma\bar{g}_{\mathrm{syn-e}}$) and contralateral inhibition ($\Sigma\bar{g}_{\mathrm{syn-i}}$). The 0 percent coherence level represents every grid point (160). The horizontal dotted lines represent the case for the isolated neuron (29). The vertical dashed lines indicate the 50 percent coherence level, and the corresponding burst-mapping diagrams are shown in figures B, D, and F.



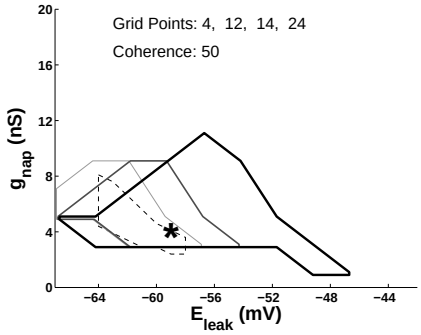
A.



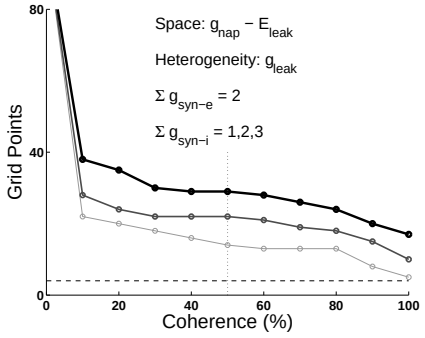
B.



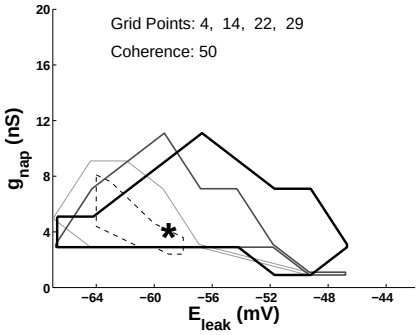
C.



D.



E.



F.

Figure 96: FPGA primary Test 5—$\bar{g}_{\mathrm{NaP}} - E_{\mathrm{Leak}}$ space using a non-canonical point with parametric and topological heterogeneity. Figures A, C, and E show the cumulative distribution of coherence measures for increasing values of ipsilateral excitation ($\Sigma\bar{g}_{\mathrm{syn-e}}$) and contralateral inhibition ($\Sigma\bar{g}_{\mathrm{syn-i}}$). The 0 percent coherence level represents every grid point (99). The horizontal dotted lines represent the case for the isolated neuron (4). The vertical dashed lines indicate the 50 percent coherence level, and the corresponding burst-mapping diagrams are shown in figures B, D, and F.

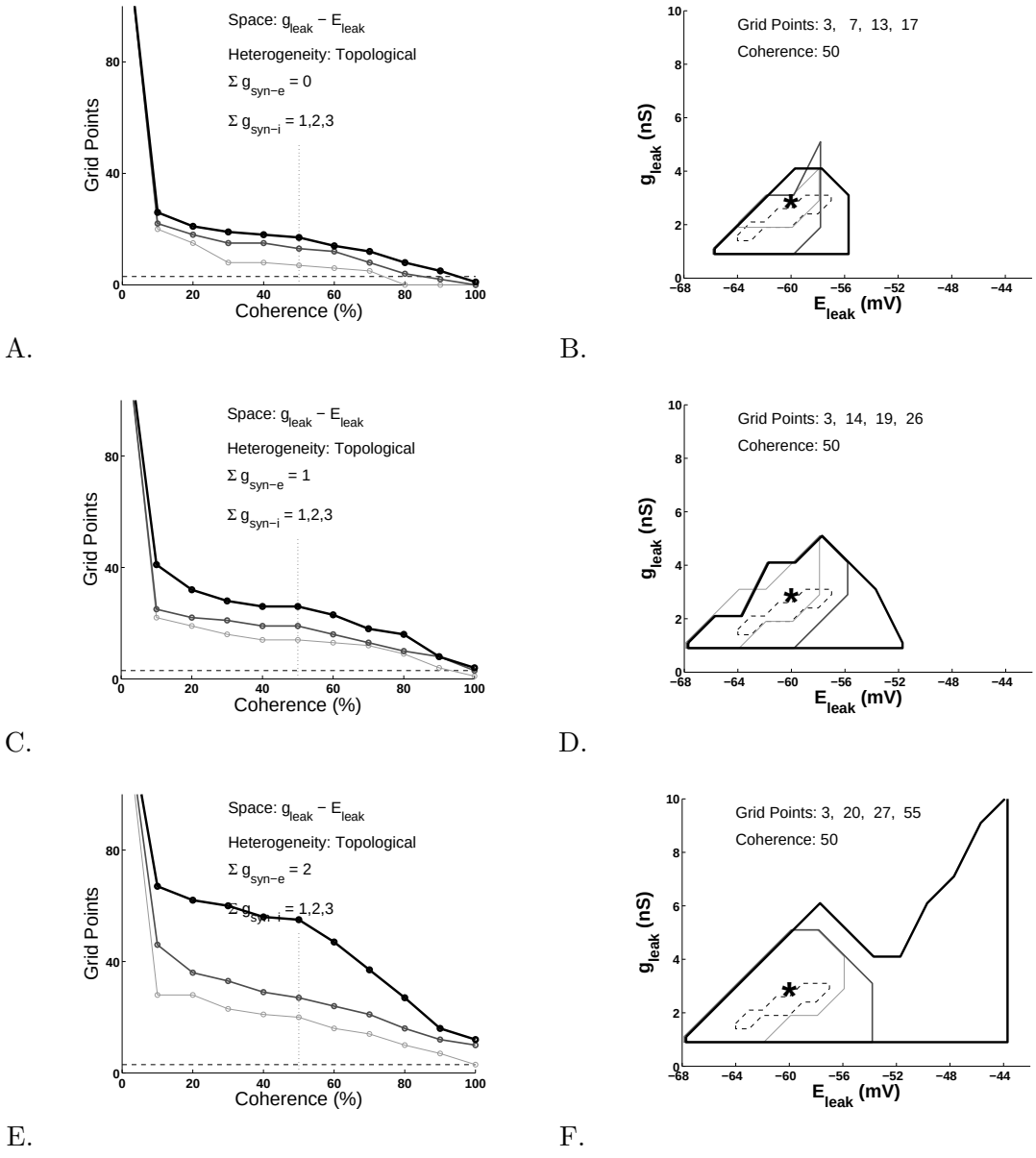


Figure 97: FPGA primary Test 5—$\bar{g}_{\mathrm{Leak}} - E_{\mathrm{Leak}}$ space using a non-canonical point with parametric and topological heterogeneity. Figures A, C, and E show the cumulative distribution of coherence measures for increasing values of ipsilateral excitation ($\Sigma\bar{g}_{\mathrm{syn-e}}$) and contralateral inhibition ($\Sigma\bar{g}_{\mathrm{syn-i}}$). The 0 percent coherence level represents every grid point (140). The horizontal dotted lines represent the case for the isolated neuron (3). The vertical dashed lines indicate the 50 percent coherence level, and the corresponding burst-mapping diagrams are shown in figures B, D, and F.

in parameter space.

We found that an all-or-nothing result was generally characteristic of homogeneous networks. In other words, to obtain 100% coherence, a homogenous network would be the best choice, but if the goal was to obtain larger bursting regions in which a smaller number of neurons could be bursting in the network, then heterogeneity should be imposed on the intrinsic parameters and smaller coherence values should be considered. This result was related to the steepness of the grid point–coherence slope, which was found to be proportional to the parametric heterogeneity imposed in the network; alternatively, for heterogeneous networks, the size of the bursting region was dependent on the coherence measure. The topological heterogeneity that we imposed on the synaptic network, however, resulted in virtually no change from the homogeneous network because the response to a distribution of synaptic values with some mean or to the same constant synaptic value for all connections will be virtually the same for both cases. Testing at the non-canonical point showed that the size of the bursting region, our metric for robustness, was not optimized at that point and was arbitrary from a modeling perspective. Finally, we showed through a variety of points in intrinsic and synaptic parameter space that the roles of the excitatory and inhibitory synaptic weights on the sizes of the burst regions were location specific.

We made an effort to minimize any bias in our results. For example, with the heterogeneity testing, we found new sets of random values for each grid point. In addition, we extended our data collection by obtaining results for three two-dimensional bursting spaces. Many of our conclusions, however, remained location specific—that is, specific to the location in intrinsic and synaptic parameter space in which we were testing. Using the stochastic gradient-descent algorithm (Section 3.1), we previously showed that a limitless number of points in parameter space could be found that replicated the single-neuron bursting dynamics. As a result, from a modeling perspective, the canonical set of parameter values was arbitrary and was certainly extremely unlikely to yield the largest bursting space. Because our measure of robustness was defined by the size of the bursting space, we obtained data for a sub-optimal point in parameter space (in fact, each of the canonical points were on the edges of the bursting spaces); finding the optimal point, however, was way beyond the

scope of our work and capabilities of our equipment. As a result, all of our findings must be qualified by the following: the results were determined at specific points in intrinsic parameter space using nine specific combinations of inhibitory and excitatory synaptic weights. Therefore, notwithstanding our attempts to minimize the bias in our results, we could not escape that location-specific bias. For example, one of the findings from primary test 1 was that the homogeneous configuration always resulted in a burst-mapping region that was bigger than the one generated by the isolated neuron, but this result was only valid for the synaptic parameters that we tested. The trend shown in Figure 74 is clear: as the excitatory coupling increases, the size of the burst-mapping space decreases. As a result, we would expect the size of the burst-mapping space for the homogeneous configuration for $\Sigma\bar{g}_{\mathrm{syn-e}} = 3$ nS to be smaller than the one for the isolated neuron.

We showed that heterogeneity increased the robustness of an HCO in the sense that a larger bursting region will result, but a coherence measure (or another similar measure) must be adopted to account for the wide variety of activity in the network. Using an analogy from digital electronics, if an AND Boolean operation is required (*i.e.*, bursting by all of the neurons or 100% coherence), then a homogeneous network is required; alternatively, if an OR Boolean operation is required (*i.e.*, bursting by as little as a single neuron or non-zero coherence), then a heterogeneous network would be beneficial. In the presence of either case, homogeneity or heterogeneity, we found large regions of parameter space in which individual, isolated neurons did not burst endogenously, but exhibited anti-phasic bursting when connected in the HCO configuration.

Because of FPGA design decisions, we only had access to the V_{mem} output from each of the neurons, and that decision limited our ability to determine the physiologically relevant burst-terminating factors for the population. The next section discusses future tests, including ones that suggest recording the slow state variable h or ionic currents. We were, however, able to determine that for the heterogeneity tests, values of the intrinsic parameter that were outside the allowable bursting range precluded bursting for that neuron—that is, considering all of the other network parameter values, recruitment to anti-phasic bursting was not possible for that neuron for that value of the intrinsic parameter. As a result,

the extreme values of the bursting range of an intrinsic parameter could be considered as the bursting *threshold* values for that parameter. In addition, we saw in previous studies discussed in Chapter 3 that burst-mapping regions were contiguous; therefore, we could argue that no *single* burst-terminating parameter exists because *all* of the model parameters synergistically work together, as opposed to a single one. Taking the argument farther, starting with the canonical point in parameter space and increasing or decreasing one of the parameters such that bursting no longer exists, a misleading claim would be to say that because the value of that one parameter is too large, bursting will not occur; this claim is misleading because some combination of the other parameters could be found to obtain bursting again. Of course, this argument assumes that a parameter value did not reach an absolute hard limit, or bursting threshold, such that bursting will always be precluded (*e.g.*, $\bar{g}_{\mathrm{Leak}} = 0$ nS).

Future Tests. The flexibility that was incorporated into the FPGA design will allow a wide variety of further testing. These future test cases fall into three categories: (1) simple extensions of our previous work, (2) implementation changes, and (3) other analysis techniques.

A simple continuation of our testing could begin with the following test cases: (1) Consider other two-dimensional intrinsic spaces from the 136 possibilities. (2) Consider other values of the excitatory and inhibitory synaptic weights, including the role of the ratio of $\Sigma\bar{g}_{\mathrm{syn-e}}$ to $\Sigma\bar{g}_{\mathrm{syn-i}}$. (3) Analyze other levels of heterogeneity. We would expect a continuity in the results such that the results from the $X\%$ heterogeneity case would fall somewhere between the $X - \Delta\%$ heterogeneity case and the $X + \Delta\%$ heterogeneity case. For example, these tests could be done to determine the spread in some heterogeneity parameter that could be tolerated to achieve perfect frequency locking (*i.e.*, the level of heterogeneity that gives a result that is clearly different from the homogeneous case). In addition, the critical level of heterogeneity, which precludes all advantages of heterogeneity, could be determined by finding where the bursting region for small coherence values is smaller than that of an isolated neuron. (4) Apply heterogeneity to other intrinsic parameters such as $\bar{g}_{\mathrm{Na}}$ and $\bar{g}_{\mathrm{K}}$. (5) Apply topological heterogeneity to the excitatory synaptic weights $\bar{g}_{\mathrm{syn-e}}$. (6) Consider

other points in intrinsic parameter space to determine more conclusively the role of location bias on our results. (7) Combine parametric and topological tests. In general, the analysis becomes more difficult when multiple system parameters are varied simultaneously, but in this case, because the topological tests gave results similar to the homogeneous tests, the results from the combination of these tests would not be difficult to differentiate.

The first set of implementation changes can be made by simply programming the synaptic weights differently to obtain a wide variety of connectivity schemes.[4] We chose a fully connected network (one in which the average, or shortest, distance between any two connected neurons is one) because we wanted to maximize the flexibility. Other connectivity schemes, including some that are deterministic and symmetrical, include a mesh (a two-dimensional mesh in which the top mesh is one half of the HCO and the bottom mesh is the other half), a nearest-neighbor network, a small-world network with subloops, or a hypercube. In addition, distant-dependent delays (*i.e.*, synaptic spread) could have also been implemented. Finally, the role of the threshold voltage of the synaptic connections (we used -10 mV; see Eq. (10)) could have been studied; previous work has shown that the network frequency is strongly determined by this threshold voltage [109]. Another set of implementation changes involves actual FPGA hardware changes. For example, because h would need to be clamped to specific values, access and control of the slow (h) intrinsic state variable would allow phase portraits in $V_{\text{mem}} - h$ space. In addition, ionic currents could also be useful for analysis. Some design issues for these changes that would need to be addressed include the sampling rate and the network size, but the tradeoff would result in more physiologically relevant explanations of the system dynamics. Regarding the network size, we began our work with a Virtex-II FPGA (10752 slices, 56 block RAMs, 56 multipliers) and soon moved to a Virtex-IV FPGA (15360 slices, 192 block RAMs, 192 multipliers), which allowed us to implement a 36-neuron HCO. We would fully expect the next-generation Virtex FPGA to have another significant increase in resources, which would allow us to test larger configurations.

[4]Technically, these are not actual hardware implementation changes but rather software changes made to the network configuration, resulting in subsets of the fully connected network.

Other analysis techniques could have also been employed. An analysis of the bursting characteristics (period, duty cycle, average spike frequency) could strengthen the results. For example, the number of neurons that are bursting at a common frequency as a function of some heterogeneity parameter or as a function of the range of natural frequencies could be shown. In addition, phase response curves (PRCs) could be used to analyze system perturbations at a variety of points in the bursting cycle; a system perturbation is analogous to a sensory, or external, input. This perturbation can be accomplished by simply bumping E_{Leak},[5] and in this sense, a system perturbation can be considered as parametric heterogeneity. For example, the network could be hit with a spike (a transient, externally applied input) at different points in the oscillation, and the response could be analyzed. For this testing, the full output, not just the peaks, would need to be available to the modeler for analysis.

[5] ΔE_{Leak} results in a constant ΔI_{Leak} that is equivalent to $\Delta E_{\text{Leak}} \times \bar{g}_{\text{Leak}}$, which is a proxy for I_{app}.

CHAPTER V

CONCLUSIONS

The goal of our work was to gain a better understanding of nature—specifically, to determine if pattern-generating networks of neurons that implemented theoretical mathematical models could absorb heterogeneous properties and make them harmonious. The human brain has a limited number of neuron types (on the order of 10^2) but significantly more neurons (on the order of 10^{10}); because the average neuron forms and receives about 10^3 synaptic connections, the number of synaptic connections is on the order of 10^{14} [62]. This provides us with the following general principle—interactions provide the complex behavior, but from very simple beginnings. Using this principle to parallel our work, we began with a well-studied model of the neuron (a conductance-based HH model) and used it to implement a fully-connected HCO using $N = 36$ neurons and N^2 connections. The complexity of the system was introduced by not only the number of connections but also the controlled forms of heterogeneity (parametric and topological). With this useful model, we were able to ask general questions about neural computation.

To understand a complex system, a researcher must first understand how each separate part works, then understand how each part interacts with those to which it is connected, and finally understand how all these local interactions combine to accomplish what that system does—as seen from the outside [85]. We accomplished these tasks by using a building-block approach starting from the bottom up—from the ion-channel level, to the neuron level, and ultimately to the population level. We took a theoretical mathematical model of the neuron, in which ion channels were described by such quantities as maximal conductances, activation states, and reversal potentials, and created an HCO in which each half of the network was replaced by a population of neurons. We chose the HCO as our platform for studying rhythmic movement because it strikes a good balance between simplicity and the rich dynamics that arise from our neurophysiologically based neuron models and because

its resulting antiphasic oscillations are well matched to control antagonistic mechanical systems. Our study began with single-neuron models in which the bursting characteristics (period, duty cycle, spike frequency) were analyzed and culminated with network models in which population statistics (network rhythmicity) were analyzed. To further our research and move closer to our goal of understanding rhythmic movement, we would also like to use these neural population models of CPGs to drive robotic limbs that use sensory feedback.

5.1 Research Contributions

At the beginning of our work, we hypothesized that the applied and natural heterogeneity imposed on a rhythmic network of neurons would result in positive changes in measures of output robustness—that is, on the sizes of the burst-mapping regions. To test this theory, we were led down many paths; in fact, scientific discovery can lead to unintended results when original hypotheses are incorrect, leading to more precise restatements. Specifically, the goal of our work was to make scientific conclusions about real biological networks such as the extent heterogeneity is functionally advantageous and the role of heterogeneity on the robust, or stable, production of rhythmic movement. The pursuit of stability, however, is a precarious one because a stable system is not necessarily a robust one—a walking robot may maintain stability over a flat, straight surface, but may become unstable from the slightest perturbation. To obtain our findings regarding the role of heterogeneity in networks of neurons (Chapter 4), we began with single-neuron models in a variety of studies (Chapter 3).

The formulation of the Hodgkin–Huxley neuron model ranks as one of the most significant conceptual breakthroughs in neuroscience. It has allowed modelers, from physiologists to computer scientists, to gain insight into fundamental neural properties that would otherwise be difficult, if not impossible to obtain. This parameterized model allows researchers to find important relationships between the parameters that comprise a vast space. The PBC model that was the focus of our work contained 17 parameters. If each parameter contained 10 values of interest, then the impossible task of evaluating 10^{17} parameter sets would need

to be completed. As a result, the modeler must either reduce the number of points of interest or smart methods must be devised to evaluate the space. We chose the latter option by creating an algorithm that combined a cost function, whose inputs were the frequency components of the neural output, with a gradient-descent approach for locating specific points in parameter space (Section 3.1). We found that the efficiency of our algorithm increased with the size of the parameter space and that the bursting space was contiguous, as opposed to a variety of islands of bursting space. These results were important because they demonstrated that we found a novel approach to studying the parameter space, which cannot be done with a brute-force method, and also resulted in an important finding about the continuity of the bursting space. Equally important was the finding that the output from the canonical set of parameters could be duplicated throughout the parameter space. These findings led us to our next set of tests.

The silicon-neuron architecture was originally designed for the HN model, but because of its versatility, simple changes to the parameter settings enabled it to function like the PBC model. This versatility was important from an engineering standpoint, and coupled with our knowledge about contiguous bursting regions, we were led to an important finding about these disparate models—that they represented points in a contiguous bursting space that spanned between the two models, from one set of parameters to another set of parameters (Section 3.2). This result was important because these models, whose bursting mechanisms relied on different ion channels, were shown to be much more closely related than previously thought. Because this study was done using the silicon implementation, the model complexity can also be viewed as a spectrum that spanned the parameter space in a parallel way in which the HN model was at one end of the spectrum (more complex) and the PBC model was at the other end (less complex); implementation difficulties increased as the model moved toward the less complex end.

At this point of our work, we determined the following critical results: (1) Bursting regions were contiguous. (2) A limitless number of points in parameter space existed that resulted in bursting dynamics that were similar to the one from the canonical point, and many more could be found for just a general simple bursting output. (3) The canonical

bursting point is arbitrary from a modeling perspective and is sub-optimal from a probability perspective. (4) The silicon model neuron proved difficult to use, and the circuitry required to generate, control, and measure heterogeneity and provide controllable synaptic weights in a fully connected network was elusive. As a result, the testing that we conducted to study the natural heterogeneity in analog circuits was limited (Section 3.3). We fabricated a population of neurons that were etched on a single integrated circuit and began our first look at intrinsic heterogeneity—a study of the built-in heterogeneity in an isolated population of analog HH models.

To better understand the role of heterogeneity in an HCO, we switched to an FPGA model neuron, and we used the size of the bursting region as the measure of robustness in the system. We quickly found that the canonical parameter set did not result in the largest bursting regions, and this important result was related to the location of the canonical parameters being on the edges of three important intrinsic bursting spaces: $\bar{g}_{\mathrm{NaP}} - \bar{g}_{\mathrm{Leak}}$, $\bar{g}_{\mathrm{NaP}} - E_{\mathrm{Leak}}$, $\bar{g}_{\mathrm{Leak}} - E_{\mathrm{Leak}}$. As much as we tried, we could not eliminate bias in our results—by definition, we were biased at the canonical parameter set. As a result, we realized that some of our findings were location specific, not necessarily general, but we still developed a few important ideas that led to the following fundamental principles of heterogeneous networks: (1) If 100% coherence is necessary, then a homogeneous network is required. (2) A coherence measure (or some similar measure) is required to evaluate population dynamics such as anti-phasic bursting. (3) Heterogeneity increases the size of the bursting region for smaller coherence values, but is not necessarily better for larger coherence values. (4) For matched intrinsic parameters, the response to a distribution of synaptic values with some mean or to the same constant synaptic value for all connections will be virtually the same for both cases. (5) The canonical parameter set does not result in the largest bursting region. These and other principles led us to our final statement: We found that heterogeneity *can* lead to more robust rhythmic networks of neurons, but the type and quantity of heterogeneity and the population-level metric that is used to analyze bursting are critical in determining *when* this occurs. As suggested at the beginning of this section, we were required to make this more precise restatement of our original hypothesis.

5.2 Research Interests

To begin our discussion of our neural implementations (Chapter 2), we described the mathematical equations that defined the conductance-based HH model, which was used for our work—the PBC neuron model (Section 2.1). We chose this model for our testing for two primary reasons: (1) At the start of this project, the silicon-neuron architecture implemented the seven-conductance HN model, and therefore, the four-conductance PBC model was seen as an effective replacement in terms of the silicon resources that would be required to implement a population of these neurons (both are intrinsic bursters, which do not require external inputs to generate bursting activity). (2) The architect of this model [8] was a faculty member of our research lab, a committee member of this work, and a co-PI on the grant that funded the majority of this work and thus would be able to provide valuable insight.

Because of the approximations that resulted from our design decisions, however, we discovered that the PBC neuron model was difficult to implement in silicon, as evidenced by the significantly reduced size of the bursting region (Section 2.2.5). In fact, the work in our research group [111] has also previously shown that model complexity and stability are positively correlated; that is, a model can more easily become unstable from a perturbation as the complexity of the model is reduced and can also be more susceptible to quantization error. We documented this last effect with the FPGA implementation during our validation of the FPGA model (Section 2.3.5). In hindsight, we may have benefited from continuing our previous silicon-neuron work that implemented the more-stable HN model [105].

In the past, our research group persisted with analog circuits, despite inherent approximations that caused difficulties with obtaining the principal scientific findings of interest [5]. In 1999, the group consisted primarily of ECE students whose research almost exclusively focused on neuromorphic engineering; attendance by multiple students at the annual summer Neuromorphic Engineering Workshop in Telluride, CO was not unusual, and many new chip designs were submitted to the bi-monthly MOSIS fabrication runs. Since that time, however, our research focus has changed considerably. As of 2006, only three of the 11 Ph.D. students were ECE students (and all three were expected to graduate in the next year), and

very few chip designs were submitted to MOSIS during the last three years (and by only one of the three ECE students). This trend does not appear likely to change because MOSIS has begun charging thousands of dollars for these same fabrication orders that were previously free and because new students are much less interested in designing analog circuits. The current students of the group have much different, and broader, research interests that range from animal experimentation to multi-electrode stimulation and recording. In fact, some students are now being co-advised by physiology professors from Emory University, which reflects the research shift to a more biological one. The bottom line is that the vision outlined by Mead almost two decades ago in his seminal book, *Analog VLSI and Neural Systems*, still has interesting opportunities to pursue, but with the current realities, the continuation of the work with the silicon neuron in our research group is unlikely.

Because of the cited difficulties with the silicon neuron and to more easily obtain the answers to the scientific questions that we posed, we were required to switch to an FPGA platform. The FPGA implementation, however, also had its share of problems, although these were comparatively easier to resolve. In addition to the quantization errors, the FPGA suffered from performance issues, but these were significantly lessened by on-chip peak detectors and FIFOs (Section 2.3.6), resulting in performance that was on the order of real time. We also found that moving to a new technology platform (a new personal computer with more RAM and a better CPU and that also included more recent versions of the Windows operating system, MATLAB, and System Generator) did not help performance; in fact, this move reduced performance by an inexplicable factor of five. Although we were able to accumulate a large amount of data, we expected to obtain more from our 24/7 testing procedures. These tests were shortened because of the following three reasons (the first one is by far the primary reason): (1) The Windows-based PC crashed, on average after about 10 hours, because of memory problems. (2) The server that stored the MATLAB scripts and data intermittently failed (sometimes because of building-wide power outages), causing our tests to stop sooner than expected. (3) Microsoft Windows automatic updates also ended our tests early in the early morning hours with the subsequent re-boot. In addition, the FPGA hardware itself was not immune to problems as one unit completely failed; a month

of testing was lost before a new FPGA, which was auspiciously ordered before the failure, arrived.

Although our implementation decisions were critical, our complementary, and arguably more important, focus was on the scientific questions in the area of rhythmic pattern-generating systems. We obtained answers to these questions using theoretical mathematical models; an alternative method, animal experimentation, would have been difficult, costly, and controversial. By directing our efforts toward the implementation of applicable technologies and modeling, we were able to present these findings involving the role of heterogeneity in rhythmic networks of neurons in our ultimate quest for a better understanding of rhythmic movement. Clearly substantially more work needs to be done in this area before researchers can truly understand the physiology that underlies this type of stereotyped movement. In the short term, because the FPGA model neuron proved to be useful for our study, it will likely be used in future research in which it is part of a system that includes robotic actuators, and this use is consistent with our original plan of using these models and results to help further the study of larger rhythmic pattern-generating systems.

APPENDIX A

MATHEMATICAL DETAILS OF THE ANALOG HETEROGENEITY CIRCUIT

The mathematical details of the analog heterogeneity circuit (Section 2.2.4) are given in the following sections.

A.1 Probability Distribution Function for the Gaussian Approximation

We obtain the probability density function, $f_{\mathbf{Y}}(y)$, for the special case where $g(x)$ is monotonically increasing, from the following formula [89]:

$$f_{\mathbf{Y}}(y) = \frac{f_{\mathbf{X}}(g^{-1}(y))}{g'(g^{-1}(y))} \tag{67}$$

where $g(x)$ is the mapping function and is defined by

$$y = g(x) = A \sinh(Bx). \tag{68}$$

Solving for x in Eq. (68) results in

$$g^{-1}(y) = x = \frac{1}{B} \sinh^{-1}(y/A). \tag{69}$$

Finding y' from Eq. (68) results in

$$y' = g'(x) = AB \cosh(Bx). \tag{70}$$

Additionally, the probability distribution function (PDF) of the uniform distribution of voltages is given by

$$f_{\mathbf{X}}(x) = \begin{cases} \frac{1}{2a} & -a \leq x \leq a \\ 0 & \text{otherwise.} \end{cases} \tag{71}$$

Rewriting Eq. (67) as

$$f_{\mathbf{Y}}(y) = \frac{f_{\mathbf{X}}(x)}{g'(x)}, \tag{72}$$

and substituting Eq. (69), Eq. (70), and Eq. (71) into Eq. (72) results in

$$f_{\mathbf{Y}}(y) = \frac{\frac{1}{2a}}{AB\cosh\left(B \cdot \frac{1}{B}\sinh^{-1}\left(y/A\right)\right)} \tag{73a}$$

$$= \frac{\frac{1}{2a}}{AB\cosh\left(\sinh^{-1}\left(y/A\right)\right)} \tag{73b}$$

$$= \frac{1}{2aAB\sqrt{(y/A)^2+1}} \tag{73c}$$

where $A\sinh\left(-aB\right) \leq y \leq A\sinh\left(aB\right)$. Note that the following hyperbolic identities

$$\cosh\left(x\right) = \frac{1}{2}(e^x + e^{-x}) \tag{74a}$$

$$\sinh^{-1}\left(x\right) = \ln\left(x + \sqrt{x^2+1}\right) \tag{74b}$$

were required to obtain the final result given in Eq. (73c) as shown in the following derivation:

$$\cosh\left(\sinh^{-1}\left(y/A\right)\right) = \frac{1}{2}\{e^{\ln\left[(y/A)+\sqrt{(y/A)^2+1}\right]} + e^{-\ln\left[(y/A)+\sqrt{(y/A)^2+1}\right]}\} \tag{75a}$$

$$= \frac{1}{2}\{[(y/A)+\sqrt{(y/A)^2+1}] + \frac{1}{(y/A)+\sqrt{(y/A)^2+1}}\} \tag{75b}$$

$$= \frac{1}{2}\{[(y/A)+\sqrt{(y/A)^2+1}] + \frac{(y/A)-\sqrt{(y/A)^2+1}}{(y/A)^2-[(y/A)^2+1]}\} \tag{75c}$$

$$= \frac{1}{2}\{[(y/A)+\sqrt{(y/A)^2+1}] - [(y/A)+\sqrt{(y/A)^2+1}]\} \tag{75d}$$

$$= \frac{1}{2}\{2\sqrt{(y/A)^2+1}\} \tag{75e}$$

$$= \sqrt{(y/A)^2+1}. \tag{75f}$$

A.2 Finding the Mapping Function for a True Gaussian Distribution

The error function, $\mathrm{erf}\left(z\right)$, is given by

$$\mathrm{erf}\left(z\right) = \frac{2}{\sqrt{\pi}}\int_0^z e^{-t^2}dt. \tag{76}$$

The Gaussian PDF is given by

$$f_{\mathbf{Y}}(y) = \frac{1}{\sigma_{\mathbf{Y}}\sqrt{2\pi}} e^{-\frac{(y-\mu_{\mathbf{Y}})^2}{2\sigma_{\mathbf{Y}}^2}}. \tag{77}$$

The Gaussian cumulative distribution function (CDF) is given by

$$F_{\mathbf{Y}}(y) = \Phi = \frac{1}{2}[1 + \text{erf}\,(\frac{y-\mu_{\mathbf{Y}}}{\sigma_{\mathbf{Y}}\sqrt{2}})]. \tag{78}$$

In general, a CDF is given by

$$F_{\mathbf{X}}(x) = P(\mathbf{X} \leq x) = \int_{-\infty}^{x} f(\alpha)d\alpha \tag{79}$$

where $0 \leq F(x) \leq 1$. Substituting Eq. (71) in Eq. (79) gives the CDF for the uniform distribution of voltages:

$$F_{\mathbf{X}}(x) = \int_{-a}^{x} \frac{1}{2a} d\alpha \tag{80a}$$

$$= \frac{1}{2} + \int_{0}^{x} \frac{1}{2a} d\alpha \tag{80b}$$

$$= \frac{1}{2}(1 + \frac{x}{a}). \tag{80c}$$

The following statistical measures also hold for Eq. (71):

$$\mu_{\mathbf{X}} = 0 \tag{81a}$$

$$\sigma_{\mathbf{X}} = \frac{1}{12}(a - (-a))^2 = \frac{1}{3}a^2. \tag{81b}$$

Note that the following relationship exists between $F_{\mathbf{X}}(x)$ and $F_{\mathbf{Y}}(y)$ (see Figure 98 for a graphical representation):

$$F_{\mathbf{X}}(x) = F_{\mathbf{Y}}(y)|_{y=g(x)} \tag{82}$$

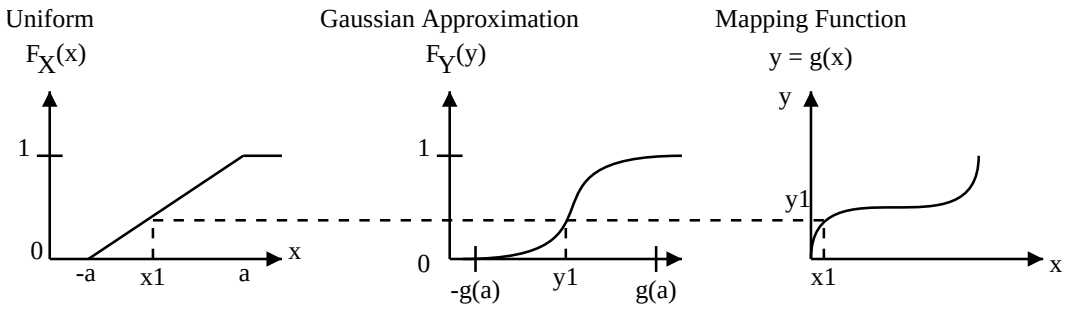


Figure 98: An alternative approach to finding the mapping function using the cumulative distribution functions.

which holds because the relationship between x and y is known. Now, using Eq. (78) and Eq. (82), we can solve for the real mapping function, $g(x) = y$, based on a true Gaussian:

$$F_{\mathbf{Y}}(y) = \Phi = \frac{1}{2}(1 + \text{erf}\,(\frac{y - \mu_{\mathbf{Y}}}{\sigma_{\mathbf{Y}}\sqrt{2}})) \tag{83a}$$

$$2F_{\mathbf{Y}}(y) - 1 = \text{erf}\,(\frac{y - \mu_{\mathbf{Y}}}{\sigma_{\mathbf{Y}}\sqrt{2}}) \tag{83b}$$

$$\text{erf}^{-1}(2F_{\mathbf{Y}}(y) - 1) = \frac{y - \mu_{\mathbf{Y}}}{\sigma_{\mathbf{Y}}\sqrt{2}} \tag{83c}$$

$$y = \mu_{\mathbf{Y}} + \sigma_{\mathbf{Y}}\sqrt{2}\,\text{erf}^{-1}\,(2F_{\mathbf{Y}}(y) - 1) \tag{83d}$$

$$y = \mu_{\mathbf{Y}} + \sigma_{\mathbf{Y}}\sqrt{2}\,\text{erf}^{-1}\,(2F_{\mathbf{X}}(x) - 1) \tag{83e}$$

$$y = \mu_{\mathbf{Y}} + \sigma_{\mathbf{Y}}\sqrt{2}\,\text{erf}^{-1}\,(-1 + 2\int_{-a}^{x} f(\alpha)d\alpha) \tag{83f}$$

$$g(x) = y = \mu_{\mathbf{Y}} + \sigma_{\mathbf{Y}}\sqrt{2}\,\text{erf}^{-1}\,(\frac{x}{a}). \tag{83g}$$

Note that the argument of erf^{-1}, where $-1 < \text{erf}^{-1}\,(x) < 1$, given in Eq. (83g) is independent of $\mu_{\mathbf{Y}}$ and $\sigma_{\mathbf{Y}}$.

A.3 Validating the Probability Density Functions

To be a valid PDF, the following must be shown for $f_{\mathbf{X}}(x)$ and $f_{\mathbf{Y}}(y)$:

$$\int_{Lx}^{Ux} f_{\mathbf{X}}(x)dx = 1 \tag{84}$$

$$\int_{Ly}^{Uy} f_{\mathbf{Y}}(y)dy = 1. \tag{85}$$

Substituting Eq. (71) in Eq. (84) gives

$$\int_{-a}^{a} f_{\mathbf{X}}(x)dx = \int_{-a}^{a} \frac{1}{2a}dx = \frac{1}{2a}(a+a) = 1. \tag{86}$$

Substituting Eq. (73c) in Eq. (85) gives

$$\int_{L}^{U} f_{\mathbf{Y}}(y)dy = \int_{A\sinh(-aB)}^{A\sinh(aB)} \frac{1}{2aAB\sqrt{(y/A)^2+1}}dy \tag{87a}$$

$$= \frac{1}{2aAB} \cdot A \cdot \sinh^{-1}(y/A)\Big|_{A\sinh(-aB)}^{A\sinh(aB)} \tag{87b}$$

$$= \frac{1}{2aB}[\sinh^{-1}(\frac{A\sinh(aB)}{A}) - \sinh^{-1}(\frac{A\sinh(-aB)}{A})] \tag{87c}$$

$$= \frac{1}{2aB}[aB + aB] = 1 \tag{87d}$$

where the relationship $\int \frac{1}{\sqrt{(y/A)^2+1}}dy = A\sinh^{-1}(y/A) + c$ was used.

A.4 Calculating the Mean

The following are three different methods for calculating the mean, $\mu_{\mathbf{Y}} = 0$:

A.4.1 Method 1

Using $f_{\mathbf{Y}}(y)$:

$$\mu_{\mathbf{Y}} = \int_{A\sinh(-aB)}^{A\sinh(aB)} y \cdot f_{\mathbf{Y}}(y)dy \tag{88a}$$

$$= \int_{A\sinh(-aB)}^{A\sinh(aB)} y \cdot \frac{1}{2aAB\sqrt{(y/A)^2+1}}dy \tag{88b}$$

$$= \frac{1}{2aAB}\int_{L'}^{U'} A^2 \frac{1}{\sqrt{w+1}}dw \tag{88c}$$

$$= \frac{A}{2aB}\int_{L'}^{U'} (w+1)^{-\frac{1}{2}}dw \tag{88d}$$

$$= \frac{A}{2aB} \cdot 2 \cdot (w+1)^{\frac{1}{2}}\Big|_{L'}^{U'} \tag{88e}$$

$$= \frac{A}{aB}\sqrt{(y/A)^2+1}\Big|_{A\sinh(-aB)}^{A\sinh(aB)} \tag{88f}$$

$$= 0 \tag{88g}$$

where $w = (y/A)^2$, $dw = \frac{1}{A^2}ydy$, and L' and U' are the translated limits of integration after making the substitution.

A.4.2 Method 2

Using $f_{\mathbf{X}}(x)$:

$$\mu_{\mathbf{Y}} = \int_{-a}^{a} y \cdot f_{\mathbf{X}}(x)dx \tag{89a}$$

$$= \int_{-a}^{a} A\sinh(Bx) \cdot \frac{1}{2a}dx \tag{89b}$$

$$= \frac{A}{2aB} \cdot \cosh(Bx)\Big|_{-a}^{a} \tag{89c}$$

$$= \frac{A}{2aB}(\cosh(aB) - \cosh(-aB)) \tag{89d}$$

$$= 0 \tag{89e}$$

since $\cosh(-x) = \cosh(x)$.

A.4.3 Method 3

Adding a zero-mean, Gaussian-distributed random variable $\mathbf{Z}$ to account for transistor mismatch results in the following stochastic mapping function:

$$\mathbf{Y} = \tilde{g}(\mathbf{X}, \mathbf{Z}) = A\sinh(B \cdot \mathbf{X}) + \mathbf{Z} \cdot I_{\text{mean}} \tag{90}$$

where $\mathbf{X}$ and $\mathbf{Z}$ are statistically independent and $I_{\text{mean}} = I_o e^{\kappa V_{\text{mean}}}$.

$$\mu_{\mathbf{Y}} = \int_{-\infty}^{\infty}\int_{-a}^{a} y \cdot f_{\mathbf{X}}(x)f_{\mathbf{Z}}(z)dxdz \tag{91a}$$

$$= \int_{-\infty}^{\infty}\int_{-a}^{a} (A\sinh(Bx) + z \cdot I_{\text{mean}}) \cdot \frac{1}{2a}f_{\mathbf{Z}}(z)dxdz \tag{91b}$$

$$= \int_{-\infty}^{\infty} \frac{1}{2a}(\frac{A}{B}\cosh(Bx) + z \cdot I_{\text{mean}} \cdot x)\Big|_{-a}^{a} f_{\mathbf{Z}}(z)dz \tag{91c}$$

$$= \int_{-\infty}^{\infty} [\frac{A}{2aB}(\cosh(aB) - \cosh(-aB)) + \frac{1}{2a} \cdot z \cdot I_{\text{mean}} \cdot (a - -a)]f_{\mathbf{Z}}(z)dz \tag{91d}$$

$$= \int_{-\infty}^{\infty} (0 + \frac{1}{2a} \cdot z \cdot I_{\text{mean}} \cdot 2a)f_{\mathbf{Z}}(z)dz \tag{91e}$$

$$= \int_{-\infty}^{\infty} (z \cdot I_{\text{mean}})f_{\mathbf{Z}}(z)dz \tag{91f}$$

$$= \mu_{\mathbf{Z}} \cdot I_{\text{mean}} \tag{91g}$$

$$= 0. \tag{91h}$$

A.5 Calculating the Standard Deviation

The following are three different methods for calculating the standard deviation, $\sigma_{\mathbf{Y}}$:

A.5.1 Method 1

Using $f_{\mathbf{Y}}(y)$:

$$\sigma_{\mathbf{Y}}^2 = \int_{A\sinh(-aB)}^{A\sinh(aB)} (y-\mu_{\mathbf{Y}})^2 \cdot f_{\mathbf{Y}}(y)dy \tag{92a}$$

where y is normalized such that $\mu_{\mathbf{Y}} = 0$. Substituting these values gives the following:

$$\sigma_{\mathbf{Y}}^2 = \int_{A\sinh(-aB)}^{A\sinh(aB)} y^2 \cdot \frac{1}{2aAB\sqrt{(y/A)^2+1}}dy \tag{92b}$$

$$= \frac{1}{2aAB}\int_{L'}^{U'} A^2w^2\frac{1}{\sqrt{w^2+1}}Adw \tag{92c}$$

where $w = y/A$, $dw = \frac{1}{A}dy$, and L' and U' are the translated limits of integration.

$$\sigma_{\mathbf{Y}}^2 = \frac{A^2}{2aB}\int_{L'}^{U'} \frac{w^2}{\sqrt{w^2+1}}dw \tag{92d}$$

$$= \frac{A^2}{2aB}(\frac{1}{2}w\sqrt{w^2+1} - \frac{\sinh^{-1}(w)}{2})\Big|_{L'}^{U'} \tag{92e}$$

because $\int \frac{y^2}{\sqrt{y^2+1}}dy = \frac{1}{2}y\sqrt{1+y^2} - \frac{\sinh^{-1}(y)}{2} + c$.

$$\sigma_{\mathbf{Y}}^2 = \frac{A^2}{2aB}(\frac{y}{2A}\sqrt{1+(y/A)^2} - \frac{\sinh^{-1}(y/A)}{2})\Big|_{A\sinh(-aB)}^{A\sinh(aB)} \tag{92f}$$

$$= \frac{A^2}{2aB}[(\frac{A\sinh(aB)}{2A}\sqrt{1+(\frac{A\sinh(aB)}{A})^2} - \frac{\sinh^{-1}(\frac{A\sinh(aB)}{A})}{2}) - (\frac{-A\sinh(aB)}{2A}\sqrt{1+(\frac{-A\sinh(aB)}{A})^2} - \frac{\sinh^{-1}(\frac{-A\sinh(aB)}{A})}{2})] \tag{92g}$$

$$= \frac{A^2}{2aB}[\sinh(aB)\sqrt{1+\sinh^2(aB)} - aB]. \tag{92h}$$

Using the identity $\cosh^2(x) - \sinh^2(x) = 1$ gives the following:

$$= \frac{A^2}{2aB}[\sinh(aB)\cosh(aB) - aB], \tag{92i}$$

and using the identity $\sinh(x)\cosh(x) = \frac{1}{2}\sinh(2x)$ gives the following:

$$= \frac{A^2}{2aB}(\frac{1}{2}\sinh(2aB) - aB) \tag{92j}$$

$$= \frac{A^2(\sinh(2aB) - 2aB)}{4aB}. \tag{92k}$$

Therefore, the standard deviation, $\sigma_{\mathbf{Y}}$, is given by the following transcendental equation of A, B, and a:

$$\sigma_{\mathbf{Y}} = \sqrt{\sigma_{\mathbf{Y}}^2} = \sqrt{\frac{A^2(\sinh(2aB) - 2aB)}{4aB}}. \tag{93}$$

A.5.2 Method 2

Using $f_{\mathbf{X}}(x)$:

$$\sigma_{\mathbf{Y}}^2 = \int_{-a}^{a} (y - \mu_{\mathbf{Y}})^2 \cdot f_{\mathbf{X}}(x) dx \tag{94a}$$

where y is normalized such that $\mu_{\mathbf{Y}} = 0$. Substituting these values gives the following:

$$\sigma^2 = \int_{-a}^{a} (A \sinh(Bx))^2 \cdot \frac{1}{2a} dx \tag{94b}$$

$$= \frac{A^2}{2a} \int_{-a}^{a} \sinh^2(Bx) dx. \tag{94c}$$

Making the substitution $\sinh^2(Bx) = \frac{\cosh(2Bx)-1}{2}$ gives the following:

$$= \frac{A^2}{2a} \int_{-a}^{a} \frac{\cosh(2Bx) - 1}{2} dx, \tag{94d}$$

and using the integral $\int \cosh(ax) dx = \frac{1}{a} \sinh(ax) + c$ gives the following:

$$= \frac{A^2}{4a} (\frac{1}{2B} \sinh(2Bx) - x)\Big|_{-a}^{a} \tag{94e}$$

$$= \frac{A^2}{4a} [(\frac{1}{2B} \sinh(2aB) - a) - (\frac{1}{2B} \sinh(-2aB) + a)] \tag{94f}$$

$$= \frac{A^2}{4a} [\frac{1}{2B} (\sinh(2aB) - \sinh(-2aB)) - 2a]. \tag{94g}$$

Making the substitution $\sinh(-x) = -\sinh(x)$ gives the following:

$$= \frac{A^2}{4a} (\frac{1}{2B} \cdot 2 \sinh(2aB) - \frac{2aB}{B}) \tag{94h}$$

$$= \frac{A^2}{4aB} (\sinh(2aB) - 2aB). \tag{94i}$$

Therefore, the standard deviation, $\sigma_{\mathbf{Y}}$, is given by the following transcendental equation of A, B, and a:

$$\sigma_{\mathbf{Y}} = \sqrt{\sigma_{\mathbf{Y}}^2} = \sqrt{\frac{A^2(\sinh(2aB) - 2aB)}{4aB}} \tag{95}$$

which is consistent with the derivation in Section A.5.1.

A.5.3 Method 3

Using $\mathbf{Y} = \tilde{g}(\mathbf{X}, \mathbf{Z})$:

$$\sigma_{\mathbf{Y}}^2 = \int_{-\infty}^{\infty} [\int_{-a}^{a} (y-\mu)^2 \cdot f_{\mathbf{X}}(x)dx] f_{\mathbf{Z}}(z)dz \tag{96a}$$

$$= \int_{-\infty}^{\infty} [\int_{-a}^{a} (A\sinh(Bx) - z \cdot I_{\text{mean}})^2 \cdot \frac{1}{2a} dx] f_{\mathbf{Z}}(z)dz \tag{96b}$$

$$= \int_{-\infty}^{\infty} [\frac{1}{2a} \int_{-a}^{a} (A^2 \sinh^2(Bx) + 2AzI_{\text{mean}} \sinh(Bx) + z^2 I_{\text{mean}}^2)dx] f_{\mathbf{Z}}(z)dz. \tag{96c}$$

The first term in the inner integrand integrates to $\frac{A^2}{4aB}(\sinh(2aB) - 2aB)$ (see Section A.5.2), the second term integrates to 0, and the third term integrates to $z^2 I_{\text{mean}}^2$. Therefore, the integral reduces to the following:

$$\sigma_{\mathbf{Y}}^2 = \int_{-\infty}^{\infty} (\frac{A^2}{4aB}(\sinh(2aB) - 2aB) + z^2 I_{\text{mean}}^2) f_{\mathbf{Z}}(z)dz \tag{96d}$$

$$= \frac{A^2}{4aB}(\sinh(2aB) - 2aB) + \sigma_{\mathbf{Z}}^2 I_{\text{mean}}^2. \tag{96e}$$

Therefore, the standard deviation, $\sigma_{\mathbf{Y}}$, is given by the following transcendental equation of A, B, a, $\sigma_{\mathbf{Z}}$, and I_{mean}:

$$\sigma_{\mathbf{Y}} = \sqrt{\sigma_{\mathbf{Y}}^2} = \sqrt{\frac{A^2(\sinh(2aB) - 2aB)}{4aB} + \sigma_{\mathbf{Z}}^2 I_{\text{mean}}^2}. \tag{97}$$

Note that $\sigma_{\mathbf{Y}}$ is dependent on $\sigma_{\mathbf{Z}}$ and not $\mu_{\mathbf{Z}}$, the mean of the Gaussian-distributed random variable $\mathbf{Z}$.

A.6 Finding the Limit of the Variance

The following are two different methods for calculating the limit of the variance, where $\lim_{a\to 0} \sigma_{\mathbf{Y}}^2 = 0$. Note that the square root can be ignored. Also, since $\lim_{a\to 0} \sigma^2 = \frac{0}{0}$, L'Hospital's rule will be used.

A.6.1 Method 1

Determine the limit for the ideal case ($\mathbf{Z} = 0$):

$$\lim_{a\to 0} \sigma_{\mathbf{Y}}^2 = \lim_{a\to 0} \frac{\frac{d}{da}[A^2(\sinh(2aB) - 2aB)]}{\frac{d}{da}[4aB]} \tag{98a}$$

$$= \lim_{a\to 0} \frac{A^2(2B\cosh(2aB) - 2B)}{4B} \tag{98b}$$

$$= \frac{0}{4B} = 0. \tag{98c}$$

A.6.2 Method 2

Determine the limit for the non-ideal case ($\mathbf{Z} \neq 0$):

$$\lim_{a\to 0} \sigma_{\mathbf{Y}}^2 = \lim_{a\to 0} \frac{\frac{d}{da}[A^2(\sinh(2aB) - 2aB) + 4aB\sigma_{\mathbf{Z}}^2 I_{\text{mean}}^2]}{\frac{d}{da}[4aB]} \tag{99a}$$

$$= \lim_{a\to 0} \frac{A^2(2B\cosh(2aB) - 2B) + 4B\sigma_{\mathbf{Z}}^2 I_{\text{mean}}^2}{4B} \tag{99b}$$

$$= \frac{4B\sigma_{\mathbf{Z}}^2 I_{\text{mean}}^2}{4B} \tag{99c}$$

$$= \sigma_{\mathbf{Z}}^2 I_{\text{mean}}^2 = 0 \tag{99d}$$

for $\sigma_{\mathbf{Z}} \to 0$.

A.7 Relationship between the Standard Deviation and a

The following is a method for determining the relationship between $\sigma_{\mathbf{Y}}$ and a, where $\frac{\partial \sigma_{\mathbf{Y}}}{\partial a} \geq 0$ ($a \geq 0$). Note that since the square root can be ignored, $\frac{\partial \sigma_{\mathbf{Y}}^2}{\partial a}$ can be found.

$$\sigma_{\mathbf{Y}}^2 = \frac{A^2}{4aB}(\sinh(2aB) - 2aB) \tag{100a}$$

$$\to \frac{\partial \sigma_{\mathbf{Y}}^2}{\partial a} = \frac{A^2}{4aB}(2B\cosh(2aB) - 2B) - \frac{A^2}{4a^2B}(\sinh(2aB) - 2aB) \tag{100b}$$

$$= \frac{A^2}{4a^2B}(2aB\cosh(2aB) - 2aB - \sinh(2aB) + 2aB) \tag{100c}$$

$$= \frac{A^2}{4a^2B}(2aB\cosh(2aB) - \sinh(2aB)). \tag{100d}$$

Note that $\frac{A^2}{4a^2B} > 0$ so we need to show that the second term is also > 0. First, note that

$$\frac{\partial(2aB\cosh(2aB) - \sinh(2aB))}{\partial a} = 4aB^2\sinh(2aB) \geq 0 \tag{100e}$$

for $a \geq 0$, and

$$(2aB\cosh(2aB) - \sinh(2aB)) = 0 \tag{100f}$$

for $a = 0$. Therefore, the second term must always be positive, and as a result,

$$\frac{\partial \sigma_{\mathbf{Y}}}{\partial a} \geq 0. \tag{100g}$$

REFERENCES

[1] Bower, J. M., "Reverse engineering the nervous system: An *in vivo*, *in vitro*, and *in computo* approach to understanding the mammalian olfactory system," in *An Introduction to Neural and Electronic Networks* (Zornetzer, S. F., Davis, J. L., and Lau, C., eds.), pp. 3–28, NY: Academic Press, 2nd ed., 1995.

[2] Bower, J. M., "What will save neuroscience?," *NeuroImage*, vol. 4, no. 3, pp. S29–S33, 1996.

[3] Bower, J. M., "Is the cerebellum sensory for motor's sake, or motor for sensory's sake: The view from the whiskers of a rat?," *Progress in Brain Research*, vol. 114, pp. 463–496, 1997.

[4] Bower, J. M. and Beeman, D., *The book of GENESIS: Exploring realistic neural models with the general neural simulation system.* Springer Verlag, 2nd ed., 1998.

[5] Bragg, J. A., *A biomorphic analog VLSI implementation of a mammalian motor unit.* PhD thesis, Georgia Institute of Technology, Atlanta, GA, 2003.

[6] Bragg, J. A., Brown, E. A., and DeWeerth, S. P., "A tunable voltage correlator," *Analog Integrated Circuits and Signal Processing*, vol. 39, pp. 89–94, Apr. 2004.

[7] Brown, T., "The intrinsic factors in the act of progression in the mammal," *Proc R Soc Lond B Biol Sci*, vol. 84, pp. 308–319, 1911.

[8] Butera, Jr., R., Rinzel, J., and Smith, J. C., "Models of respiratory rhythm generation in the Pre-Bötzinger complex. I. Bursting pacemaker neurons," *Journal of Neurophysiology*, vol. 82, no. 1, pp. 382–397, 1999.

[9] Butera, Jr., R., Rinzel, J., and Smith, J. C., "Models of respiratory rhythm generation in the Pre-Bötzinger complex. II. Populations of coupled pacemaker neurons," *Journal of Neurophysiology*, vol. 82, no. 1, pp. 398–415, 1999.

[10] Calabrese, R. L., "Neural coordination: Taking the lead from a model," *Current Biology*, vol. 9, pp. R680–R683, Sep 1999.

[11] Cavagna, G., "The sources of external work in level walking and running," *Journal of Physiology (London)*, vol. 262, pp. 639–657, 1976.

[12] Cavagna, G., "Mechanical work in terrestrial locomotion: Two basic mechanisms for minimizing energy expenditure," *American Journal of Physiology*, vol. 233, pp. 243–261, 1977.

[13] Chow, C. C., "Phase-locking in weakly heterogeneous neuronal networks," *Physica D*, vol. 118, pp. 343–370, Jul 1998.

[14] Connor, J. and Stevens, C., "Voltage clamp studies of a transient outward membrane current in gastropod neural somata," *Journal of Physiology - London*, vol. 213, no. 1, pp. 21–30, 1971.

[15] Cymbalyuk, G. S., Patel, G. N., Calabrese, R. L., DeWeerth, S. P., and Cohen, A. H., "Modeling alternation to synchrony with inhibitory coupling: A neuromorphic VLSI approach," *Neural Computation*, vol. 12, no. 10, pp. 2259–2278, 2000.

[16] Darghouth, N., "Modeling A-current modulation in an identified molluscan neuron," Master's thesis, Georgia Institute of Technology, Atlanta, GA, 2004.

[17] Del Negro, C., Johnson, S., Butera, Jr., R., and Smith, J., "Models of respiratory rhythm generation in the Pre-Bötzinger complex. III. Experimental tests of model predictions," *Journal of Neurophysiology*, vol. 86, pp. 59–74, Jul 2001.

[18] Del Negro, C., Koshiya, N., Butera, Jr., R., and Smith, J., "Persistant sodium current, membrane properties and bursting behavior of Pre-Bötzinger complex inspiratory neurons *in vitro*," *Journal of Neurophysiology*, vol. 88, pp. 2242–2250, Nov 2002.

[19] DeWeerth, S., Patel, G., and Simoni, M., "Variable linear-range subthreshold OTA," *Electronics Letters, IEE*, vol. 33, no. 15, pp. 1309–1311, 1997.

[20] DeWeerth, S. P., "VLSI chip modeled after a leech," *IEEE Computational Science & Engineering*, vol. 2, no. 4, p. 83, 1995.

[21] DeWeerth, S. P., Nielsen, L., Mead, C. A., and Astrom, K. J., "A simple neuron servo," *IEEE Transactions on Neural Networks*, vol. 2, no. 2, pp. 248–251, 1991.

[22] DeWeerth, S. P., Reid, M. S., Brown, E. A., and Butera, Jr., R. J., "A comparative analysis of multi-conductance neuronal models *in Silico*," *Biological Cybernetics*, 2006. Article in press.

[23] Diorio, C., "Floating-gate MOSFETs," in *Analog VLSI: Circuits and Principles* (Liu, S.-C., Kramer, J., Indiveri, G., Delbrück, T., and Douglas, R., eds.), ch. 4, pp. 93–120, The MIT Press, 2002.

[24] Douglas, R. and Mahowald, M., "Design and fabrication of analog VLSI neurons," in *Methods in neuronal modeling: From ions to networks* (Koch, C. and Segev, I., eds.), ch. 8, pp. 293–312, Cambridge, MA: The MIT Press, 1998.

[25] Duong, Q.-H., Duong, H.-N., Nguyen, T.-K., and Lee, S.-G., "All CMOS current-mode exponential function generator," in *ICACT*, pp. 528–531, IEEE, 2004.

[26] Dupeyron, D., Le Masson, S., Deval, Y., Le Masson, G., and Dom, J., "A BiCMOS implementation of the Hodgkin–Huxley formalism," in *Proceedings of the 5th International conference of Microelectronics for Neural Networks and Fuzzy Systems*, pp. 311–316, IEEE Computer Society Press, 1996.

[27] Eisen, J. and Marder, E., "Mechanisms underlying pattern generation in lobster stomatogastric ganglion as determined by selective inactivation of identified neurons. III. synaptic connections of electrically coupled pyloric neurons," *Journal of Neurophysiology*, vol. 48, pp. 1392–1415, 1982.

[28] El Mourabit, A., Lu, G.-N., and Pittet, P., "Wide-linear-range subthreshold OTA for low-power, low-voltage, and low-frequency applications," *IEEE Transactions on Circuits and Systems—I: Regular Papers*, vol. 52, pp. 1481–1488, Aug 2005.

[29] Farley, C. and Ko, T., "External mechanical power output in lizard locomotion," *Journal of Experimental Biology*, vol. 200, pp. 2177–2188, 1997.

[30] Farquhar, E., "A biologically inspired silicon neuron," Master's thesis, Georgia Institute of Technology, Atlanta, GA, 2003.

[31] Farquhar, E. and Hasler, P., "A bio-physically inspired silicon neuron," *IEEE Transactions on Circuits and Systems—I: Regular Papers*, vol. 52, pp. 477–488, Mar 2005.

[32] Fish, A., Milrud, V., and Yadid-Pecht, O., "High-speed and high-precision current winner-take-all circuit," *IEEE Transactions on Circuits and Systems—II: Express Briefs*, vol. 52, pp. 131–135, Mar 2005.

[33] Foster, W., Ungar, L., and Schwaber, J., "Significance of conductances in Hodgkin-Huxley models," *Journal of Neurophysiology*, vol. 70, pp. 2502–2518, Dec 1993.

[34] Full, R. and Farley, C., "Musculoskeletal dynamics in rhythmic systems - a comparative approach to legged locomotion," in *Biological and Neural Control of Posture and Movement* (Winters, J. and Crago, P., eds.), New York, NY: Springer-Verlag, 2000.

[35] Gabbiani, F. and Koch, C., "Principles of spike train analysis," in *Methods in neuronal modeling: From ions to networks* (Koch, C. and Segev, I., eds.), ch. 9, pp. 313–360, Cambridge, MA: The MIT Press, 1998.

[36] Gerstner, W., "Spiking neurons," in *Pulsed Neural Networks* (Maass, W. and Bishop, C., eds.), Cambridge, MA: The MIT Press, 1999.

[37] Getting, P. A., "Mechanisms of pattern generation underlying swimming in *Tritonia*. III. Intrinsic and synaptic mechanisms for delayed excitation," *Journal of Neurophysiology*, vol. 49, pp. 1036–1050, 1983.

[38] Gilbert, B., "Current-mode circuits from a translinear viewpoint: A tutorial," in *Analogue IC design: The current-mode approach* (Toumazou, C., Lidgey, F., and Haigh, D., eds.), ch. 2, pp. 11–91, London, UK: Peter Peregrinus Ltd., 1993.

[39] Golowasch, J., Abbott, L., and Marder, E., "Activity-dependent regulation of potassium currents in an identified neuron of the stomatogastric ganglion of the crab *Cancer borealis*," *Journal of Neuroscience*, vol. 19, no. 20, pp. A1–A5, 1999.

[40] Golowasch, J., Goldman, M. S., Abbott, L., and Marder, E., "Failure of averaging in the construction of a conductance-based neuron model," *Journal of Neurophysiology*, vol. 87, pp. 1129–1131, 2002.

[41] Graas, E., Brown, E., and Lee, R. H., "An FPGA-based approach to high-speed simulation of conductance-based neuron models," *Neuroinformatics*, vol. 2, no. 4, pp. 417–436, 2004.

[42] Gray, C., "Synchronous oscillations in neuronal systems: Mechanisms and functions," *Journal of Computational Neuroscience*, vol. 1, pp. 11–38, 1994.

[43] Gray, P. R. and Meyer, R. G., *Analysis and design of analog integrated circuits*. John Wiley & Sons, Inc., 3rd ed., 1993.

[44] Grimm, K. and Sauer, A. E., "The high number of neurons contributes to the robustness of the locust flight-CPG against parameter variation," *Biological Cybernetics*, vol. 72, no. 4, pp. 329–335, 1995.

[45] Hahnloser, R., Sarpeshkar, R., Mahowald, M., Douglas, R., and Seung, H., "Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit," *Nature*, vol. 405, no. 6789, pp. 947–951, 2000.

[46] Harrison, R. R., Bragg, J. A., Hasler, P., Minch, B. A., and DeWeerth, S. P., "A CMOS programmable analog memory-cell array using floating-gate circuits," *IEEE Transactions on Circuits and Systems-II*, vol. 48, pp. 4–11, Jan 2001.

[47] Hastings, A., *The art of analog layout*. Upper Saddle River, NJ: Prentice Hall, 2001.

[48] Hatsopoulos, N., "Coupling the neural and physical dynamics in rhythmic movements," *Neural Computation*, vol. 8, no. 3, pp. 567–581, 1996.

[49] Hatsopoulos, N. and Warren, W., "Resonance tuning in rhythmic arm movements," *Journal of Motor Behavior*, vol. 28, no. 1, pp. 3–14, 1996.

[50] Hill, A., Hooser, S. V., and Calbrese, R., "Half-center oscillators underlying rhythmic movements," in *The handbook of brain theory and neural networks* (Arbib, M., ed.), pp. 507–510, The MIT Press, 2002.

[51] Hill, A., Lu, J., Masino, M., Ølsen, O., and Calabrese, R., "A model of a segmental oscillator in the leech heartbeat neuronal network," *Journal of Computational Neuroscience*, vol. 10, no. 3, pp. 281–302, 2001.

[52] Hille, B., *Ion channels of excitable membranes*. Sunderland, MA: Sinauer Associates, Inc., 3rd ed., 2001.

[53] Himmelbauer, W., Furth, P., Pouliquen, P., and Andreou, A., "Log domain filters in subthreshold MOS," tech. rep., Johns Hopkins University, 1996.

[54] Hines, M. and Carnevale, N., "The NEURON simulation environment," *Neural Computation*, vol. 9, no. 6, pp. 1179–1209, 1997.

[55] Hodgkin, A. and Huxley, A., "A quantitative description of membrane current and its application to conduction and excitation in nerve," *Journal of Physiology (London)*, vol. 117, pp. 500–544, 1952.

[56] Horiuchi, T. K., Morris, T. G., Koch, C., and DeWeerth, S. P., "Analog VLSI circuits for attention-based, visual tracking," in *Advances in Neural Information Processing Systems 9: Proceedings of the 1996 Conference* (Mozer, M. C., Jordan, M. I., and Petsche, T., eds.), pp. 706–712, The MIT Press, 1997.

[57] Hudson, T. and DeWeerth, S., "Implementation of motoneuronal behavior using analog VLSI circuits," in *42nd Midwest Symposium on Circuits and Systems*, (Piscataway, NJ), IEEE Press, 1999.

[58] Ichikawa, T., "Synchronous firing dynamics in a heterogeneous neurosecretory-cell population in an insect," *Brain Research*, vol. 929, pp. 156–165, Mar 2002.

[59] Izhikevich, E. M., "Neural excitability, spiking and bursting," *International Journal of Bifurcation and Chaos*, vol. 10, no. 6, pp. 1171–1266, 2000.

[60] Jin, C. T., Leong, P. H., and Leong, M., "An FPGA-based electronic cochlea," *Eurasip Journal on Applied Signal Processing*, vol. 7, pp. 629–638, Jul 2003.

[61] Johnston, D. and Wu, S. M., *Foundations of cellular neurophysiology*. Cambridge, MA: The MIT Press, 1997.

[62] Kandel, E. R., Schwartz, J. H., and Jessell, T. M., eds., *Principles of neural science*. New York, NY: McGraw-Hill/Appleton & Lange, 4th ed., 2000.

[63] Karbowski, J. and Ermentrout, G. B., "Synchrony arising from a balanced synaptic plasticity in a network of heterogeneous neural oscillators," *Physical Review E*, vol. 65, Mar 2002.

[64] Kitano, H., "Biological robustness," *Nature Review Genetics*, vol. 5, pp. 826–837, Nov 2004.

[65] Kiyatkin, E. and Rebec, G., "Heterogeneity of ventral tegmental area neurons: Single-unit recording and iontophoresis in awake, unrestrained rats," *Neuroscience*, vol. 85, pp. 1285–1309, Aug 1998.

[66] Korkmaz, P., Akgul, B., Palem, K., and Chakrapani, L., "Advocating noise as an agent for ultra-low energy computing: Probabilistic complementary metal-oxide-semiconductor devices and their characteristics," *Japanese Journal of Applied Physics Part I—Regular Papers Brief Communications & Review Papers*, vol. 45, pp. 3307–3316, Apr 2006.

[67] Kuramoto, Y., "Lecture notes in physics," in *Proceedings of the International Symposium on Mathematical Problems in Theoretical Physics* (Araki, H., ed.), vol. 39, Springer, 1975.

[68] Laurent, G. and Davidowitz, H., "Encoding of olfactory informatin with oscillating neural assemblies," *Science*, vol. 265, pp. 1872–1875, 1994.

[69] Laurent, S., Masson, C., and Jakob, I., "Whole-cell recording from honeybee olfactory receptor neurons: Ionic currents, membrane excitability and odourant response in developing workerbee and drone," *European Journal of Neuroscience*, vol. 15, pp. 1139–1152, Apr 2002.

[70] Lewis, M. A., Hartmann, M. J., Etienne-Cummings, R., and Cohen, A. H., "Control of a robot leg with an adaptive aVLSI CPG chip," *Neurocomputing*, vol. 38, pp. 1409–1421, Jun 2001.

[71] Linares-Barranco, B., Sanchez-Sinencio, E., Rodriguez-Vazquez, A., and Huertas, J., "A CMOS implementation of Fitzhugh-Nagumo neuron model," *IEEE Journal of Solid-State Circuits*, vol. 26, pp. 956–965, Jul 1991.

[72] Lisman, J. and Idiart, M., "Storage of 7 ± 2 short-term memories in oscillatory subcycles," *Science*, vol. 267, pp. 1512–1515, 1995.

[73] Liu, S.-C., Kramer, J., Indiveri, G., Delbrück, T., and Douglas, R., *Analog VLSI: Circuits and principles*. Cambridge, MA: The MIT Press, 2002.

[74] Lyon, R. F. and Mead, C. A., "An analog electronic cochlea," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, no. 7, pp. 1119–1134, 1988.

[75] MacKay-Lyons, M., "Central pattern generation of locomotion: A review of the evidence," *Physical Therapy*, vol. 82, pp. 69–83, Jan 2002.

[76] Mahowald, M. A. and Douglas, R., "A silicon neuron," *Nature*, vol. 354, pp. 515–518, Dec 1991.

[77] Mahowald, M. A. and Mead, C., "The silicon retina," *Scientific American*, vol. 264, no. 5, pp. 76–82, 1991.

[78] Mak, T., Rachmuth, G., Lam, K. P., and Poon, C.-S., "Field programmable gate array implementation of neuronal ion channel dynamics," *Neural Engineering, 2005. Conference Proceedings. 2nd International IEEE EMBS Conference on*, pp. 144–148, 2005.

[79] Marder, E. and Calabrese, R. L., "Principles of rhythmic motor pattern generation," *Physiological Reviews*, vol. 76, pp. 687–717, Jul 1996.

[80] Marder, E. and Prinz, A. A., "Modeling stability in neuron and network function: The role of activity in homeostasis," *BioEssays*, vol. 24, no. 12, pp. 1145–1154, 2002.

[81] Mead, C. A., *Analog VLSI and neural systems*. Reading, MA: Addison-Wesley, 1989.

[82] Meyer-Bäse, U. and Scheich, H., "Artificial implementation of auditory neurons: A comparison of biologically motivated models and a new transfer function oriented model," *Biological Cybernetics*, vol. 77, pp. 123–130, Aug 1997.

[83] Miller, J. and Selverston, A., "Mechanisms underlying pattern generation in lobster stomatogastric ganglion as determined by selective inactivation of identified neurons. IV. Network properties of pyloric system," *Journal of Neurophysiology*, vol. 48, no. 6, pp. 1416–1432, 1982.

[84] Miller, P., Brody, C. D., Romo, R., and Wang, X.-J., "A recurrent network model of somatosensory parametric working memory in the prefrontal cortex," *Cerebral Cortex*, vol. 13, pp. 1208–1218, Nov 2003.

[85] Minsky, M., ed., *Society of mind*. Simon & Schuster, 1988.

[86] Nadim, F., Ølsen, O., Schutter, E. D., and Calabrese, R., "Modeling the leech heartbeat elemental oscillator: I. Interactions of intrinsic and synaptic currents," *Journal of Computational Neuroscience*, vol. 2, no. 3, pp. 215–235, 1995.

[87] Ølsen, O., Nadim, F., and Calabrese, R., "Modeling the leech heartbeat elemental oscillator: II. Exploring the parameter space," *Journal of Computational Neuroscience*, vol. 2, no. 3, pp. 237–257, 1995.

[88] Oppenheim, A. V., Schafer, R. W., and Buck, J. R., *Discrete-time signal processing*. Upper Saddle River, NJ: Prentice Hall, 2nd ed., 1999.

[89] Papoulis, A. and Pillai, S. U., *Probability, random variables, and stochastic processes with errata sheet*. McGraw-Hill Science/Engineering/Math, 4th ed., 2001.

[90] Patel, G. and DeWeerth, S., "Analogue VLSI Morris-Lecar neuron," *Electronics Letters*, vol. 33, no. 12, pp. 997–998, 1997.

[91] Patel, G., Brown, E., and DeWeerth, S. P., "A neuromorphic VLSI system for modeling the neural control of axial locomotion," in *Advances in Neural Information Processing Systems 12: Proceedings of the 1999 Conference* (Solla, S. A., Leen, T. K., and Müller, K.-R., eds.), pp. 724–730, The MIT Press, 2000.

[92] Patel, G. N., Holleman, J. H., and DeWeerth, S. P., "Analog VLSI model of intersegmental coordination with nearest-neighbor coupling," in *Advances in Neural Information Processing Systems 10: Proceedings of the 1997 Conference* (Jordan, M. I., Kearns, M. J., and Solla, S. A., eds.), pp. 719–725, The MIT Press, 1998.

[93] Pearson, K., "Neural adaptation in the generation of rhythmic behavior," *Annual Review of Physiology*, vol. 62, pp. 723–753, 2000.

[94] Prinz, A. A., Billimoria, C. P., and Marder, E., "Alternative to hand-tuning conductance-based models: Construction and analysis of databases of model neurons," *Journal of Neurophysiology*, vol. 90, pp. 3998–4015, 2003.

[95] Rasche, C. and Douglas, R., "An improved silicon neuron," *Analog Integrated circuits and Signal Processing*, vol. 23, no. 3, pp. 227–236, 2000.

[96] Reid, M. S., Brown, E. A., and DeWeerth, S. P., "Subthreshold CMOS array for generating a Gaussian distribution of currents," *IEEE Transactions on Circuits and Systems—II: Express Briefs*, vol. 53, pp. 1123–1127, Oct 2006.

[97] Reid, M. S., Brown, E. A., and DeWeerth, S. P., "A parameter-space search algorithm tested on a Hodgkin–Huxley model," *Biological Cybernetics*, 2007. Submitted.

[98] Saraga, F. and Skinner, F., "Dynamics and diversity in interneurons: A model exploration with slowly inactivating potassium currents," *Neuroscience*, vol. 113, no. 1, pp. 193–203, 2002.

[99] Sargeni, F. and Bonaiuto, V., "Digitally programmable nonlinear function generator for neural networks," *Electronics Letters*, vol. 41, no. 3, pp. 143–145, 2005.

[100] Sarpeshkar, R., "Analog versus digital: Extrapolating from electronics to neurobiology," *Neural Computation*, vol. 10, no. 7, pp. 1601–1638, 1998.

[101] Schild, J. and Kunze, D., "Experimental and modeling study of Na+ current heterogeneity in rat nodose neurons and its impact on neuronal discharge," *Journal of Neurophysiology*, vol. 6, no. 78, pp. 3198–3209, 1997.

[102] Segev, I., Fleshman, Jr., J., and Burke, R., "Computer simulation of group Ia EPSPs using morphologically realistic models of cat alpha-motoneurons," *Journal of Neurophysiology*, vol. 64, no. 2, pp. 648–660, 1990.

[103] Selverston, A., Rabinovich, M., Arbanel, H., Elson, R., Szucs, A., Pinto, R., Huerta, R., and Varona, P., "Reliable circuits from irregular neurons: A dynamical approach to understanding central pattern generators," *Journal of Physiology - Paris*, vol. 94, no. 5–6, pp. 357–374, 2000.

[104] Serra-Graells, F. and Huertas, J. L., "Low-voltage CMOS subthreshold log-domain filtering," *IEEE Transactions on Circuits and Systems—I: Regular Papers*, vol. 52, pp. 2090–2100, Oct 2005.

[105] Simoni, M. F., *Synthesis and analysis of a physical model of biological rhythmic motor control with sensorimotor feedback*. PhD thesis, Georgia Institute of Technology, Atlanta, GA, 2002.

[106] Simoni, M. F., Cymbaluyk, G. S., Sorensen, M. E., Calabrese, R. L., and DeWeerth, S. P., "A multiconductance silicon neuron with biologically matched dynamics," *IEEE Transactions on Biomedical Engineering*, vol. 51, pp. 342–354, Feb 2004.

[107] Simoni, M. F., Cymbaluyk, G. S., Sorensen, M. Q., Calabrese, R. L., and DeWeerth, S. P., "Development of hybrid systems: Interfacing a silicon neuron to a leech heart interneuron," in *Advances in Neural Information Processing Systems 13: Proceedings of the 2000 Conference* (Leen, T., Dietterich, T., and Tresp, V., eds.), The MIT Press, 2001.

[108] Simoni, M. F., Patel, G. N., and DeWeerth, S. P., "Analog VLSI model of the leech heartbeat elemental oscillator," in *Sixth Annual Computational Neuroscience Meeting*, (Big Sky, MT), 1997.

[109] Skinner, F. K., Kopell, N., and Marder, E., "Mechanisms for oscillation and frequency control in reciprocally inhibitory model neural networks," *Journal of Computational Neuroscience*, vol. 1, no. 1–2, pp. 69–87, 1994.

[110] Sorensen, M., Calabrese, R., and DeWeerth., S., "The functional role of model complexity in a neuronal oscillator," in *2004 Abstract Viewer/Itinerary Planner*, (Washington, DC), Society for Neuroscience, 2004. Program No. 420.6.

[111] Sorensen, M. E., *Functional consequences of model complexity in hybrid neural-microelectronic systems*. PhD thesis, Georgia Institute of Technology, Atlanta, GA, 2005.

[112] Tiesinga, P. and José, J. V., "Robust gamma oscillations in networks of inhibitory hippocampal interneurons," *Network-Computation in Neural Systems*, vol. 11, pp. 1–23, Feb 2000.

[113] Tsallis, C. and Stariolo, D. A., "Generalized simulated annealing," *Physica A*, vol. 233, pp. 395–406, Nov 1996.

[114] Tsodyks, M., Mitkov, I., and Sompolinsky, H., "Pattern of synchrony in inhomogeneous networks of oscillators with pulse interactions," *Physical Review Letters*, vol. 71, pp. 1280–1283, Aug 1993.

[115] Uyemura, J. P., *CMOS logic circuit design*. Kluwer Academic Publishers, 1999.

[116] Valdés, J. J., Belanche, L. A., and Alquézar, R., "Fuzzy heterogeneous neurons for imprecise classification problems," *International Journal of Intelligent Systems*, vol. 15, pp. 265–276, Mar 2000.

[117] Venance, L. and Glowinski, J., "Heterogeneity of spike frequency adaptation among medium spiny neurones from the rat striatum," *Neuroscience*, vol. 122, no. 1, pp. 77–92, 2003.

[118] Vetter, P., Roth, A., and Hausser, M., "Propagation of action potentials in dendrites depends on dendritic morphology," *Journal of Neurophysiology*, vol. 85, no. 2, pp. 926–937, 2001.

[119] Wallén, P. and Grillner, S., "Fictive locomotion in the lamprey spinal cord *in vitro* compared with swimming in the intact and spinal animal," *Journal of Physiology - London*, vol. 37, pp. 225–239, 1987.

[120] Wang, X.-J. and Buzsáki, G., "Gamma oscillation by synaptic inhibition in a hippocampal interneuronal network model," *Journal of Neuroscience*, vol. 16, pp. 6402–6413, Oct 1996.

[121] Weinstein, R. K. and Lee, R. H., "Architectures for high-performance FPGA implementations of neural models," *Journal of Neural Engineering*, vol. 3, pp. 21–34, Mar 2006.

[122] Weinstein, R. K., Reid, M. S., and Lee, R. H., "Methodology and design flow for assisted neural model implementations in FPGAs," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, pp. 1–1, 2007. Article in press.

[123] Weinstein, R. and Lee, R., "Computationally intensive motoneuron modeling in FPGAs," in *2004 Abstract Viewer/Itinerary Planner*, (Washington, DC), Society for Neuroscience, 2004. Program No. 921.5.

[124] Weinstein, R. and Lee, R., "Design of high performance physiologically-complex motoneuron models in FPGAs," in *Neural Engineering, 2005. Conference Proceedings. 2nd International IEEE EMBS Conference on*, (Washington, DC), pp. 526–528, 2005.

[125] Weisstein, E. W., "Hypergeometric distribution." MathWorld—A Wolfram Web Resource, May 2005. http://mathworld.wolfram.com/HypergeometricDistribution.html.

[126] Weisstein, E. W., "Normal distribution." MathWorld—A Wolfram Web Resource, May 2005. http://mathworld.wolfram.com/NormalDistribution.html.

[127] Weisstein, E. W., "Taylor series." MathWorld—A Wolfram Web Resource, May 2005. http://mathworld.wolfram.com/TaylorSeries.html.

[128] Weste, N. H. E. and Eshraghian, K., *Principles of CMOS VLSI design: A systems perspective*. Reading, MA: Addison-Wesley, 2nd ed., 1993.

[129] White, J. A., Chow, C. C., Ritt, J., Soto-Treviño, C., and Kopell, N., "Synchronization and oscillatory dynamics in heterogeneous, mutually inhibited neurons," *Journal of Computational Neuroscience*, vol. 5, pp. 5–16, 1998.

[130] Wiesenfeld, K., Colet, P., and Strogatz, S. H., "Frequency locking in Josephson arrays: Connection with the Kuramoto model," *Physical Review E*, vol. 57, pp. 1563–1569, 1998.

[131] Williams, T., Grillner, S., Smoljaninov, V., Wallén, P., Kashin, S., and Rossignol, S., "Locomotion in lamprey and trout: The relative timing of activation and movement," *Journal of Experimental Biology*, vol. 143, pp. 559–566, 1989.

[132] Wolpert, D. and Macready, W., "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, pp. 67–82, Apr 1997.

[133] Zelenin, P. V., Grillner, S., Orlovsky, G. N., and Deliagina, T. G., "Heterogeneity of the population of command neurons in the lamprey," *Journal of Neuroscience*, vol. 21, pp. 7793–7803, Oct 2001.

VITA

Mike Reid was born in Camden, NJ in 1966 and moved to North Palm Beach, FL a few years later. He attended Palm Beach County public schools from kindergarten through high school. After high school graduation, he attended Auburn University in Alabama and earned a Bachelor of Electrical Engineering in 1988. Upon graduation, he worked for Pratt & Whitney Aircraft in West Palm Beach, FL until 1992. He then attended Carnegie Mellon University in Pittsburgh, PA and earned a Masters in Business Administration in 1994. Upon graduation, he worked on Wall Street for Bankers Trust Company, whose location was directly across from the World Trade Center in lower Manhattan, until 1999. He then attended the Georgia Institute of Technology in Atlanta, GA and completed the requirements for a Ph.D. in Electrical and Computer Engineering in December 2006 and subsequently graduated in May 2007. He currently lives in Atlanta, GA with his wife, Lucy, his two boys, Michael and Steven, and his two dogs, Einstein and Eleanor.